Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IF184802

PENGENALAN PELAT NOMOR KENDARAAN PADA DATA VIDEO MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK BERDASARKAN CITRA SINTETIS SEBAGAI DATA LATIH

GHIFAROZA RAHMADIANA
NRP 05111640000057

Dosen Pembimbing I
Dini Adni Navastara, S.Kom., M.Sc.

Dosen Pembimbing II
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – IF184802

PENGENALAN PELAT NOMOR KENDARAAN PADA DATA VIDEO MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK BERDASARKAN CITRA SINTETIS SEBAGAI DATA LATIH

GHIFAROZA RAHMADIANA
NRP 05111640000057

Dosen Pembimbing I
Dini Adni Navastara, S.Kom., M.Sc.

Dosen Pembimbing II
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – IF184802

VIDEO BASED LICENSE PLATE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK TRAINED ON SYNTHETIC IMAGES

GHIFAROZA RAHMADIANA
NRP 05111640000057

First Advisor
Dini Adni Navastara, S.Kom., M.Sc.

Second Advisor
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Department of Informatics
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PENGENALAN PELAT NOMOR KENDARAAN PADA DATA VIDEO MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK BERDASARKAN CITRA SINTETIS SEBAGAI DATA LATIH

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
GHIFAROZA RAHMADIANA
NRP: 05111640000057

Disetujui oleh Pembimbing Tugas Akhir:

1. Dini Adni Navastara, S.Kom., M.Sc. .....
   (NIP. 19851017 201504 2 001) (Pembimbing 1)
2. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. .....
   (NIP. 19751220 200112 2 002) (Pembimbing 2)

SURABAYA
Januari, 2020

(Halaman ini sengaja dikosongkan)

PENGENALAN PELAT NOMOR KENDARAAN PADA DATA VIDEO MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK BERDASARKAN CITRA SINTETIS SEBAGAI DATA LATIH

Nama Mahasiswa	**: Ghifaroza Rahmadiana**
NRP	**: 05111640000057**
Departemen	**: Teknik Informatika, FTEIC-ITS**
Dosen Pembimbing 1	**: Dini Adni Navastara, S.Kom., M.Sc.**
Dosen Pembimbing 2	**: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**

ABSTRAK

Teknologi License Plate Recognition (LPR) atau pengenalan pelat nomor telah diadopsi di banyak aplikasi lalu lintas modern, seperti tempat parkir dan pemantauan lalu lintas. Metode deep learning berbasis Convolutional Neural Network (CNN) telah menunjukkan kemajuan yang luar biasa pada bidang LPR. Namun, dalam proses pelatihan CNN dibutuhkan sampel berlabel dalam jumlah besar, sedangkan untuk memperoleh sampel bisa sangat sulit karena pelabelan yang memakan waktu dan biaya tinggi. Salah satu solusi untuk mengatasi permasalahan tersebut adalah menggunakan sampel sintetis untuk melatih CNN.

Pada Tugas Akhir ini akan dibuat sistem LPR yang menggunakan citra sintetis sebagai data latih. Sistem LPR terdiri dari tiga tahapan utama yaitu deteksi area pelat nomor menggunakan Single Shot Detector (SSD), segmentasi karakter pada area pelat nomor yang berhasil dideteksi menggunakan Maximally Stable Extremal Regions, dan pengenalan karakter yang berhasil disegmentasi menggunakan CNN. Adapun arsitektur jaringan SSD akan dilatih menggunakan citra pelat sintetis berspesifikasi standar pelat nomor Indonesia sedangkan arsitektur jaringan CNN akan dilatih menggunakan citra karakter sintetis.

Pengujian dilakukan dalam empat skenario yaitu pengujian kondisi pada citra pelat sintetis, pengujian optimizer CNN, pengujian pada data video, dan pengujian pada data gambar. Hasil uji coba optimal didapatkan dengan menonaktifkan kondisi oklusi pada dataset pelat sintetis untuk melatih arsitektur SSD dan menggunakan optimizer Adam pada arsitektur CNN dengan nilai akurasi pengenalan sebesar 88,92% pada data video menggunakan majority vote dan 86,77% pada data gambar. Rata-rata total waktu yang dibutuhkan untuk deteksi area, segmentasi karakter, dan pengenalan karakter pada satu area pelat dalam satu frame data video adalah 39,44ms.

Kata kunci: Citra sintetis, *Convolutional Neural Network*, Pengenalan pelat nomor kendaraan, *Single Shot Detector*

VIDEO BASED LICENSE PLATE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK TRAINED ON SYNTHETIC IMAGES

Student's Name : Ghifaroza Rahmadiana
Student's ID : 05111640000057
Department : Informatics, Faculty of ELECTICS-ITS
First Advisor : Dini Adni Navastara, S.Kom., M.Sc.
Second Advisor : Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

ABSTRACT

License Plate Recognition (LPR) technology has been adopted in many modern traffic applications, such as parking lots and traffic monitoring. Convolutional Neural Network (CNN) based deep learning technology has shown remarkable progress in the LPR field. However, in the CNN training process it takes a large number of labeled samples, while obtaining samples can be difficult because of the time-consuming and high cost of labeling. One solution to overcome this problem is to use synthetic samples to train CNN.

In this undergraduate thesis an LPR system will be made using synthetic images as the train data. The LPR consists of three stages, license plate area detection using Single Shot Detector (SSD), character segmentation using Maximally Stable Extremal Regions, and character recognition using CNN. The SSD network architecture will be trained using synthetic plate images with Indonesian plate standard specifications, while the CNN network architecture will be trained using synthetic character images.

The test is carried out in four scenarios, namely testing the condition on synthetic plate images, testing the CNN optimizer, testing on video data, and testing on image data. Optimal test results are obtained by disabling occlusion conditions on synthetic license plates images to train SSD architecture and using Adam optimizer on CNN architecture resulting recognition accuracy value of 88.92% in video data using majority vote and 86.77% in

image data. The average total time needed to detect area, character segmentation, and character recognition in one plate area in one frame in video data is 39.44ms.

Keywords: ***Convolutional Neural Network, License Plate Recognition, Synthetic Image, Single Shot Detector***

KATA PENGANTAR

Puji syukur penulis sampaikan kepada Tuhan yang Maha Esa karena berkat rahmat-Nya penulis dapat melaksanakan Tugas Akhir yang berjudul:

"**PENGENALAN PELAT NOMOR KENDARAAN PADA DATA VIDEO MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK BERDASARKAN CITRA SINTETIS SEBAGAI DATA LATIH**"

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan banyak pihak, oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan kepada:

1. Orangtua dan keluarga penulis, yang telah memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Dini Adni Navastara, S.Kom., M.Sc. dan Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku pembimbing I dan II yang telah membimbing, memberikan motivasi, dan masukan dalam menyelesaikan Tugas Akhir ini.
3. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku Kepala Departemen Teknik Informatika ITS dan seluruh dosen dan karyawan Departemen Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah.
4. Muhammad Adistya Azhar dan Nuzha Musyafira yang telah menemani, mendukung, dan membantu penulis selama perkuliahan hingga semester 7 termasuk dalam pengerjaan Tugas Akhir ini.
5. Teman-teman admin Laboratorium Pemrograman, Inan, Chael, Mala, Frandita, Teja, John, Bagas, Setya, Komang, Famus, Mbak Bonbon, dan Mas Raca, yang selalu

memberi dukungan moral selama pengerjaan Tugas Akhir ini.

6. Teman-teman admin Laboratorium Komputasi Cerdas & Visi (KCV) yang telah menyediakan tempat yang membantu penulis dalam mengerjakan Tugas Akhir ini.
7. Seluruh mahasiswa Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama menjalani masa kuliah di Teknik Informatika ITS.
8. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa laporan Tugas Akhir ini masih jauh dari kata sempurna. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan penulis kedepannya. Selain itu, penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum.

Surabaya, Januari 2020

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

DAFTAR KODE SUMBER

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi pengenalan pelat nomor kendaraan atau *License Plate Recognition* (LPR) telah diadopsi di banyak aplikasi lalu lintas modern, seperti tempat parkir, pemantauan lalu lintas, dan kontrol keamanan jalanan. Baru-baru ini, metode *deep learning* berbasis *Convolutional Neural Network* (CNN) telah menunjukkan kemajuan yang luar biasa dalam bidang visi komputer, seperti klasifikasi objek, deteksi objek, dan segmentasi citra. Beberapa penelitian LPR yang telah dilakukan adalah [1] dan [2] dimana keduanya membuktikan bahwa CNN memiliki performa yang baik walaupun masih terbatas pada skenario tertentu disebabkan dataset yang digunakan jumlahnya tidak banyak karena dataset dianotasi dan diambil secara manual dari citra pelat di dunia nyata.

Dalam proses pelatihan CNN dibutuhkan sampel berlabel dalam jumlah besar dengan distribusi variabel yang mewakili varians intra-kelas dari kondisi dunia nyata, sedangkan untuk memperoleh sampel saja bisa jadi sangat sulit karena adanya peraturan-peraturan yang mengatur privasi. Bahkan jika jumlah sampel yang cukup dapat diperoleh, pelabelan sampel tersebut akan memakan waktu serta biaya yang tinggi.

Salah satu solusi yang sudah ada untuk mengatasi permasalahan tersebut adalah sistem LPR yang dirancang menggunakan CNN dengan sampel sintetis untuk menghindari pengumpulan dan pelabelan ribuan citra yang diperlukan untuk melatih CNN secara manual [3]. Dalam menghasilkan citra pelat sintetis, perlu memperhitungkan *key variables* yang dibutuhkan untuk memodelkan berbagai kondisi yang memenuhi aspek-aspek citra pelat yang asli. Sistem tersebut diuji menggunakan dataset citra pelat kendaraan asli dari tiga negara yang berbeda (Italia, Cina, dan Taiwan). Hasilnya LPR yang dilatih menggunakan citra pelat sintetis tersebut mengungguli beberapa sistem LPR lainnya yang dilatih menggunakan citra pelat asli, menunjukkan bahwa

citra sintetis efektif dalam melatih CNN untuk LPR jika citra sintetis tersebut memiliki varians yang cukup dari *key variables* yang memenuhi aspek-aspek citra pelat yang asli.

Berdasarkan hal-hal di atas, pada Tugas Akhir ini akan diimplementasikan arsitektur CNN untuk pengenalan pelat nomor kendaraan pada data video dimana data latihnya berupa citra pelat dan karakter sintetis. Adapun citra pelat dan karakter sintetis yang dibuat memiliki spesifikasi pelat nomor kendaraan bermotor standar Indonesia.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana membuat citra pelat sintetis yang dapat memenuhi aspek-aspek citra pelat asli?
2. Bagaimana cara melakukan deteksi area pelat nomor pada kendaraan dengan citra pelat sintetis sebagai data latih?
3. Bagaimana cara melakukan segmentasi karakter pada area pelat yang berhasil dideteksi?
4. Bagaimana cara melakukan pengenalan karakter pada pelat nomor menggunakan CNN dengan citra karakter sintetis sebagai data latih?
5. Bagaimana performa sistem *License Plate Recognition* menggunakan CNN dengan citra sintetis sebagai data latih?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Implementasi tugas akhir ini menggunakan bahasa pemrograman Python 3.
2. Dataset pelatihan merupakan citra pelat dan karakter sintetis dengan spesifikasi pelat nomor kendaraan sepeda motor standar Indonesia.

3. Dataset untuk pengujian merupakan data video yang berisi pelat nomor kendaraan sepeda motor standar Indonesia yang posisinya tepat lurus menghadap kamera CCTV.
4. Dataset hanya berisi satu pelat nomor kendaraan.
5. *Background* untuk pembuatan citra pelat sintetis menggunakan MIRFLICKR-1M dataset.
6. Karakter pada pelat nomor yang dapat dikenali adalah 26 karakter huruf (A-Z) dan 10 karakter angka (0-9).

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah melakukan deteksi dan pengenalan karakter pada pelat nomor kendaraan menggunakan *Convolutional Neural Network* (CNN) dengan citra sintetis sebagai data latih.

1.5 Manfaat

Tugas akhir ini diharapkan menjadi sistem yang mampu memudahkan pengontrolan pelat nomor kendaraan dalam sistem keamanan parkir dengan memanfaatkan teknologi *License Plate Recognition* (LPR).

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir yang berisi pendahuluan, deskripsi dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri dari latar belakang diajukannya Tugas Akhir, rumusan masalah dan batasan masalah yang ditetapkan, serta manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Terdapat pula subbab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian literatur berupa jurnal yang digunakan sebagai referensi untuk pengerjaan Tugas Akhir ini. Literatur yang dipelajari pada pengerjaan Tugas Akhir ini berasal dari jurnal ilmiah yang diambil dari berbagai sumber di internet, beserta berbagai literatur *online* tambahan terkait *Convolutional Neural Network*, TensorFlow dan Keras.

1.6.3 Implementasi Perangkat Lunak

Pada tahap ini akan dilaksanakan implementasi metode dan algoritma yang telah direncanakan. Implementasi sistem menggunakan Python 3 sebagai bahasa pemrograman, TensorFlow dan Keras sebagai *framework*, serta *library* pendukung lainnya.

1.6.4 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi dilakukan untuk mengetahui hasil dan performa algoritma yang telah diimplementasikan. Dataset yang digunakan dalam proses pengujian berupa data video berisi pelat nomor kendaraan bermotor standar Indonesia yang diambil dari *Closed-Circuit TeleVision* (CCTV) di area parkir Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember, Surabaya. Pada tahap evaluasi akan dilakukan evaluasi pada model CNN dari sisi akurasi, presisi, *recall*, dan *f1-score*.

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

Bab I Pendahuluan

Bab ini berisi penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

Bab II Tinjauan Pustaka

Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang *Convolutional Neural Network* dan *library* yang digunakan.

Bab III Perancangan Sistem

Bab ini berisi pembahasan mengenai perancangan dari metode *Convolutional Neural Network* yang digunakan untuk pengenalan pelat nomor kendaraan pada data video.

Bab IV Implementasi

Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode sumber yang digunakan dalam proses implementasi.

Bab V Uji Coba dan Evaluasi

Bab ini membahas tahapan uji coba, kemudian hasil uji coba dievaluasi terhadap kinerja dari sistem yang dibangun.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini membahas mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut mengenai citra, sintesis citra, pengolahan citra digital, *Convolutional Neural Network*, dan beberapa teori lain yang mendukung pembuatan Tugas Akhir. Penjelasan ini bertujuan untuk memberikan gambaran umum dan diharapkan dapat mendukung sistem yang dibangun.

2.1 Citra

Secara umum pengertian citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra sebagai keluaran suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpanan [4].

Citra secara umum terbagi menjadi dua bagian, yaitu citra analog dan citra digital. Citra analog merupakan citra yang bersifat kontinu, seperti gambar pada monitor televisi, lukisan, dan lain sebagainya. Citra digital merupakan representasi dari sebuah citra dua dimensi sebagai sebuah kumpulan nilai digital yang disebut elemen gambar atau *pixel*. *Pixel* adalah satuan terkecil dari citra yang mengandung nilai terkuantisasi yang mewakili kecerahan dari sebuah warna pada sebuah titik tertentu.

2.2 Sintesis Citra

Sintesis citra merupakan proses menciptakan citra dari citra lain atau data non-citra. Sintesis citra dilakukan ketika sebuah citra yang diinginkan secara fisik tidak mungkin, tidak praktis untuk diperoleh, atau tidak ada dalam bentuk fisik sama sekali. Ada dua bentuk sintesis citra. Bentuk yang pertama adalah rekonstruksi dari suatu citra dengan menggunakan beberapa gambar proyeksi. Bentuk yang kedua adalah visualisasi. Visualisasi adalah operasi yang menciptakan citra untuk keperluan presentasi yang tidak

mungkin didasarkan pada benda-benda fisik sama sekali [5]. Pada Tugas Akhir akan dibuat citra pelat dan karakter sintetis yang akan digunakan sebagai data latih. Contoh citra asli dan citra sintetis dapat dilihat pada Gambar 2.1.

Gambar 2.1 (a) Citra pelat asli; (b) Citra pelat sintetis.

2.3 Pengolahan Citra Digital

Pengolahan citra digital adalah ilmu yang mempelajari hal-hal yang berkaitan dengan perbaikan kualitas gambar, transformasi gambar, melakukan pemilihan citra ciri, melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkandung pada citra, dan melakukan kompresi atau reduksi data. Input dari pengolahan citra adalah citra, sedangkan outputnya adalah citra hasil pengolahan. Pengolahan citra adalah pemrosesan citra, dengan maksud untuk mendapatkan kualitas citra yang diinginkan [6]. Beberapa operasi pengolahan citra untuk membuat citra pelat dan karakter sintetis yang akan diimplementasikan dalam Tugas Akhir ini adalah penambahan *noise*, pengaburan, rotasi, pemotongan (*cropping*), dan penskalaan (*scaling*).

2.3.1 Penambahan Noise

Pada saat proses *capture* (pengambilan gambar), beberapa gangguan mungkin terjadi, seperti kamera tidak fokus atau munculnya bintik-bintik yang bisa jadi disebabkan oleh proses *capture* yang tidak sempurna. Setiap gangguan pada citra disebut *noise* [7]. Pada Tugas Akhir ini akan digunakan *Gaussian noise* untuk menambahkan *noise* pada citra sintetis.

Gaussian noise merupakan model *noise* yang mengikuti distribusi normal standar dengan rata-rata nol dan standar deviasi

1. Efek dari *noise* ini pada citra adalah munculnya titik-titik berwarna yang jumlahnya sama dengan persentase *noise* [7]. *Gaussian noise* dapat dilihat pada (2.1).

$$f(i,j) = g(i,j) + p.a \tag{2.1}$$

Dimana:

$f(i,j)$ = nilai citra dengan *noise*
$g(i,j)$ = nilai citra asli
a = nilai bilangan acak berdistribusi *gaussian*
p = persentase *noise*

Contoh citra sebelum dan sesudah ditambah *Gaussian noise* dapat dilihat pada Gambar 2.2.

Gambar 2.2 Citra sebelum (kiri) dan sesudah ditambah *Gaussian noise* (kanan).

2.3.2 Pengaburan Citra

Pengaburan (*blurring*) yaitu filter spasial *low-pass* yang melenyapkan detil halus dari suatu citra. Pengaburan dicapai melalui konvolusi dari seluruh koefisien *mask* bernilai sama, semakin besar ukuran *mask* maka makin besar efek pengaburan [8].

Pada Tugas Akhir ini akan digunakan filter *Gaussian* untuk mengaplikasikan efek pengaburan pada citra sintetis. Filter *Gaussian* merupakan filter linier yang memilih nilai pembobotan untuk setiap *pixel* dipilih berdasarkan fungsi *Gaussian*. Filter *Gaussian* dapat digunakan untuk mengurangi *noise* yang ada pada citra digital. Filter *Gaussian* didapat dari operasi konvolusi yaitu operasi perkalian yang dilakukan antara matriks kernel dengan

matriks citra asli [9]. Matriks kernel *Gausian* didapat dari fungsi komputasi dari distribusi *Gaussian*, seperti yang dapat dilihat pada (2.2).

$$G(i,j) = c.e\frac{(i-u)^2 + (j-v)^2}{2\sigma^2} \quad (2.2)$$

Dimana:
c dan σ = konstanta
$G(i,j)$ = elemen matriks kernel *Gaussian* pada posisi (i, j)
(u,v) = indeks tengah dari matriks kernel *Gaussian*

Sedangkan untuk perkalian antara bobot matriks gambar asli dengan bobot matriks kernel *Gaussian* dapat dilihat pada (2.3).

$$\frac{1}{K}\sum_{p=0}^{N-1}\left(\sum_{q=0}^{M-1} G(p,q)Pixel\ A\left(1+p-\frac{(N-1)}{2}, i+q-\frac{(M-1)}{2}\right)\right) \quad (2.3)$$

Dimana:
Pixel A = citra A (citra asli)
Pixel B(i, j) = bobot hasil perkalian pada posisi (i, j)
N = jumlah kolom matriks kernel
M = jumlah baris matriks kernel
K = penjumlahan semua bobot di G
G(p, q) = elemen matriks kernel *Gauss* pada posisi (p, q)

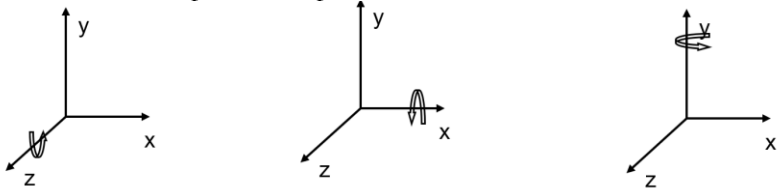
Citra yang akan diproses dibagi menjadi dua jenis *pixel*, yaitu *pixel* batas dan *pixel* dalam. *Pixel* batas yaitu *pixel* yang berada pada bagian terluar citra. *Pixel* dalam yaitu *pixel* yang berada di dalam *pixel* batas. Untuk *pixel* yang berada di dalam, perkalian dilakukan menggunakan rumus pertama yaitu mencari nilai *pixel* baru sebagai *pixel* tengah dan bobotnya dikalikan dengan bobot pada *pixel* tengah matriks kernel. Hasil dari perkalian tersebut dijumlahkan dengan hasil perkalian antara bobot *pixel* yang ada pada tetangganya dengan bobot dari *pixel* pada matriks kernel. Untuk *pixel* yang berada pada sudut perbatasan, sebelum melakukan perkalian harus dicari bobot pada *pixel* luar (*dummy*). Bobot *pixel* ini dicari dengan menggunakan metode interpolasi

yaitu melihat kedua *pixel* yang berdekatan juga searah (horizontal atau vertikal). Jika bobot *pixel* kurang dari 0 maka bobot tersebut akan dijadkan 0. Apabila ada *pixel* yang memiliki bobot lebih besar dari 255 maka bobotnya dijadikan 255. Contoh citra sebelum dan sesudah ditambah efek pengaburan dengan filter *Gaussian* dapat dilihat pada Gambar 2.3.

Gambar 2.3 (a) Citra sebelum ditambah efek pengaburan dengan filter *Gaussian*; (b) Citra sesudah ditambah efek pengaburan dengan filter *Gaussian.*

2.3.3 Rotasi Citra

Rotasi merupakan suatu transformasi geometri memindahkan nilai-nilai *pixel* dari posisi awal menuju ke posisi akhir yang ditentukan melalui nilai variabel rotasi [10]. Pada Tugas Akhir ini, rotasi citra akan digunakan untuk proses pembuatan citra pelat dan karakter sintetis yang akan dirotasi pada sumbu x sebesar θ^o, sumbu y sebesar ϕ^o, dan sumbu z sebesar ψ^o dengan matriks transformasi $T_x, T_y,$ dan T_z [11] yang dapat dilihat pada (2.4). Ilustrasi rotasi dapat dilihat pada Gambar 2.4.



Gambar 2.4 Ilustrasi rotasi citra [11]

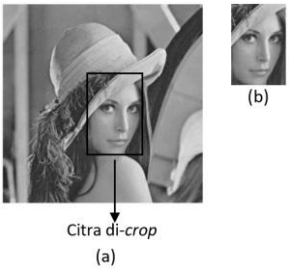
$$[T_x] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & sin\theta & 0 \\ 0 & -sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[T_y] = \begin{bmatrix} cos\phi & 0 & -sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ sin\phi & 0 & cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

$$[T_z] = \begin{bmatrix} cos\psi & sin\psi & 0 & 0 \\ -sin\psi & cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3.4 Pemotongan Citra (*cropping*)

Cropping adalah memotong satu bagian dari citra sehingga diperoleh citra yang berukuran lebih kecil. Proses ini bertujuan untuk memisahkan objek yang satu dengan objek yang lain dalam suatu gambar [10]. Contoh citra sebelum dan sesudah di-*crop* dapat dilihat pada Gambar 2.5.



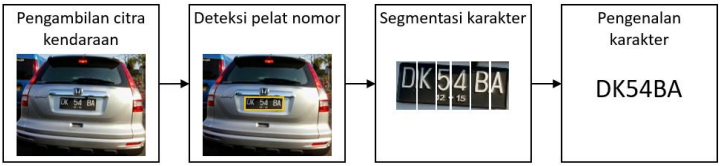
Gambar 2.5 (a) Citra sebelum sesudah di-*crop*; (b) Citra sesudah sesudah di-*crop*.

2.4 Pengenalan Pelat Nomor Kendaraan

Dalam beberapa tahun terakhir, *License Plate Recognition* (LPR) atau pengenalan pelat nomor kendaraan menjadi salah satu

pendekatan yang berguna untuk pengamatan kendaraan yang diterapkan dalam rangka memenuhi beberapa tujuan seperti penegakan keselamatan lalu lintas, pengumpulan teks tol otomatis, dan sistem parkir kendaraan.

Algoritma LPR umumnya dibagi dalam empat tahap yaitu pengambilan citra kendaraan, deteksi pelat nomor, segmentasi karakter, dan pengenalan karakter seperti yang dapat dilihat pada Gambar 2.6. Langkah pertama yaitu pengambilan citra kendaraan yang terlihat mudah tetapi sebenarnya cukup penting karena sulit untuk menangkap citra kendaraan yang bergerak secara *real time* sehingga tidak ada komponen kendaraan terutama pelat nomor yang terlewat. Keberhasilan langkah keempat bergantung pada bagaimana langkah kedua dan ketiga dapat mendeteksi pelat nomor kendaraan dan memisahkan setiap karakter yang ada di dalamnya. Sebagian besar sistem LPR didasarkan pada pendekatan umum seperti *Artificial Neural Network* (ANN), *Probabilistic Neural Network* (PNN), *Optical Character Recognition* (OCR), *Sliding Concentrating Window* (SCW), *Support Vector Machine* (SVM), dan lain sebagainya [12].

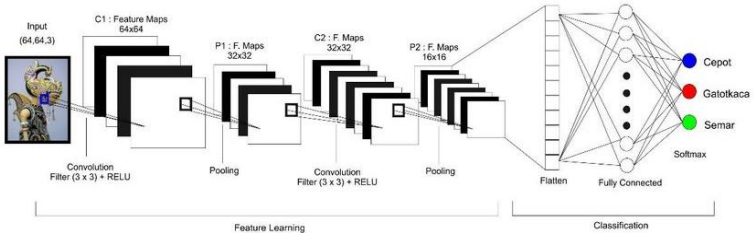


Gambar 2.6 Tahapan Umum pada Pengenalan Pelat Nomor Kendaraan

2.5 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu algoritma dari *deep learning* yang merupakan pengembangan dari *Multi Layer Percepton* (MLP) yang dirancang untuk mengolah data dalam bentuk dua dimensi, misalnya gambar atau suara. CNN sering digunakan untuk mengenali citra benda atau pemandangan, melakukan deteksi dan segmentasi objek [13].

Penelitian awal yang mendasari penemuan ini dilakukan oleh Hubel dan Wiesel [14] yang melakukan penelitian visual korteks pada indera penglihatan kucing. Penelitian ini sangat berguna dalam sistem pemrosesan visual yang pernah ada. Hingga banyak penelitian yang terinspirasi dari cara kerjanya dan menghasilkan model-model baru. Arsitektur dari CNN dibagi menjadi dua bagian besar, *extraction layer* dan *classification layer* [15], seperti yang dapat dilihat pada Gambar 2.7.



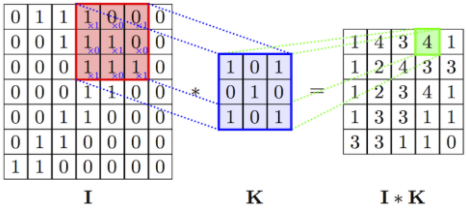
Gambar 2.7 Contoh arsitektur *Convolutional Neural Network* [16]

Pada *extraction* layer terjadi proses penerjemahan dari sebuah citra menjadi *feature map* berisi angka-angka yang merepresentasikan citra tersebut. Bagian ini terdiri dari *convolutional layer* dan *pooling layer*. *Feature map* yang dihasilkan dari *convolutional layers* masih berbentuk *array* multidimensi, sehingga harus diubah menjadi sebuah *feature vector* agar bisa digunakan sebagai masukan untuk *fully connected layer* pada *classification layer*. *Fully connected layer* yang dimaksud disini adalah MLP yang memiliki beberapa *hidden layer, activation function, output layer*, dan *loss function.*

2.5.1 Convolution Layer

Convolution Layer melakukan operasi konvolusi pada output dari lapisan sebelumnya. Konvolusi adalah istilah matematis yang artinya mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Tujuan dilakukan konvolusi pada data citra adalah untuk mengekstrak fitur dari citra masukan [13]. Ilustrasi cara kerja konvolusi bisa dilihat pada Gambar 2.8,

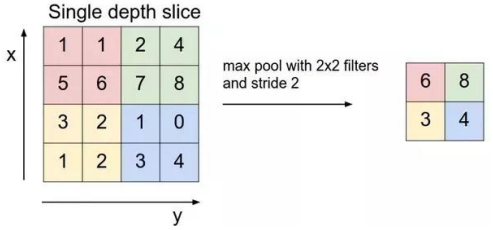
dimana I adalah citra, K adalah *filter* atau *kernel* yang digunakan, I * K adalah hasil operasi konvolusi.



Gambar 2.8 Ilustrasi cara kerja konvolusi [17]

2.5.2 Pooling Layer

Fungsi dari *Pooling Layer* adalah mereduksi ukuran dari data. Terdapat beberapa tipe *Pooling Layer* diantaranya yaitu *max*, *average*, *sum* dan lainnya. Metode *Pooling* dalam CNN yang biasa digunakan adalah *Max Pooling* & *Average Pooling*. *Max Pooling* membagi *output* dari *Convolution Layer* menjadi beberapa matriks kecil lalu mengambil nilai maksimal dari tiap matriks untuk menyusun matriks citra yang telah direduksi, sedangkan *Average Pooling* akan memilih nilai rata-ratanya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun obyek citra mengalami translasi. Ilustrasi cara kerja *Max Pooling* bisa dilihat pada Gambar 2.9.



Gambar 2.9 Ilustrasi cara kerja *Max Pooling* [18]

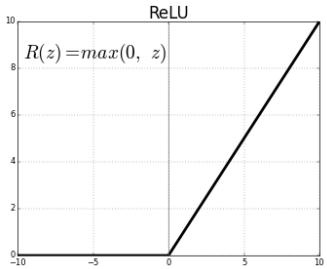
2.5.3 Fully Connected Layer

Fully Connected Layer dalam penerapannya sama dengan *Multi Layer Perceptron* (MLP) yang bertujuan untuk melakukan

transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. *Feature map* dari *Convolution Layer* perlu ditransformasi menjadi data satu dimensi terlebih dahulu yang disebut *feature vector* sebelum dapat dimasukkan ke dalam sebuah *Fully Connected Layer*. Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak reversibel, *Fully Connected Layer* diimplementasikan di akhir jaringan [19].

2.5.4 ReLU Activation Function

Fungsi aktivasi befungsi untuk menentukan apakah neuron tersebut harus aktif atau tidak berdasarkan nilai masukan. Salah satu contoh fungsi aktivasi adalah ReLU (Rectified Linear Unit) dimana fungsi ini melakukan *thresholding* dengan nilai nol terhadap nilai masukan, dimana seluruh nilai yang kurang dari nol akan dijadikan nol, seperti pada Gambar 2.10.



Gambar 2.10 ReLU *Activation Function* [20]

2.5.5 Fungsi Softmax

Fungsi *softmax* biasa digunakan dalam klasifikasi banyak kelas. *Softmax* memberikan nilai probabilitas untuk setiap label kelas, dimana jumlah seluruh probabilitas adalah 1. *Softmax* pada dasarnya adalah probabilitas eksponensial yang dinormalisasi dari nilai masukan sejumlah kelas pada model klasifikasi seperti pada (2.5). Operasi akan menghasilkan nilai probabilitas. Label dari data masukan akan ditentukan berdasarkan kelas dengan nilai probabilitas tertinggi.

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{J} e^{y_j}} \quad (2.5)$$

Dimana:

S = probabilitas input tersebut merupakan kelas i
y = persamaan linear fungsi pre-aktivasi
i = salah satu kelas dari J kelas yang ada

2.5.6 Cross Entropy

Loss function merupakan fungsi yang menggambarkan kerugian yang dihasilkan oleh model. *Loss function* dikatakan baik, ketika menghasilkan *error* yang diharapkan paling rendah. Pada permasalahan klasifikasi banyak kelas, *cross entropy* adalah *loss function* yang biasa digunakan. *Cross entropy* akan menghitung *error* antara nilai prediksi dengan nilai sebenarnya, seperti pada (2.6). Selanjutnya, nilai *error* akhir diambil dari rata-rata hasil *cross entropy*, seperti pada (2.7).

$$D(S_i, T_i) = -\sum_j T_{ij} \log S_{ij} \quad (2.6)$$

$$J(W, b) = \frac{1}{n} \sum_i D(S_i, T_i) \quad (2.7)$$

Dimana:
S = nilai prediksi
T = nilai sebenarnya

2.5.7 Stochastic Gradient Descent

Ketika melatih sebuah model, dibutuhkan sebuah *loss function* yang dapat mengukur kualitas dari setiap bobot atau parameter tertentu. *Stochastic Gradient Descent* (SGD) adalah algoritma pengoptimalan. Tujuan pengoptimalan adalah untuk menemukan parameter yang dapat meminimalkan nilai *error* dari *loss function*. SGD adalah algoritma yang digunakan untuk memperbarui nilai bobot dan bias pada neuron di *neural network*. Pada dasarnya operasi yang dilakukan hanya mengurangi bobot

awal dengan sebagian nilai dari nilai gradien yang sudah kita dapat. Nilai sebagian disini diwakili oleh parameter bernama *learning rate*, seperti yang terlihat pada (2.8) dan (2.9).

$$w_{j+1} = w_j - \alpha \frac{\partial}{\partial w_j} J(W, b) \tag{2.8}$$

$$b_{j+1} = w_j - \alpha \frac{\partial}{\partial b_j} J(W, b) \tag{2.9}$$

Dimana:
α = initial learning rate
J = cost function

2.5.8 Adagrad

Adagrad adalah algoritma pengoptimalan berbasis gradien yang memperbarui *learning rate* setiap parameternya. Adagrad adalah algoritma yang digunakan untuk memperbarui nilai bobot dan bias pada neuron di *neural network*. Adagrad menggunakan *adaptive learning rate* untuk setiap parameter. Berbeda dengan *Stochastic Gradient Descent* (SGD) yang selalu menggunakan *learning rate* yang sama, Adagrad memiliki sebuah *learning rate* untuk setiap parameter dan secara terpisah beradaptasi saat proses pelatihan. Pembaruan dilakukan untuk tiap parameter θ_j dengan gradien *loss function* g_j, seperti yang dapat dilihat pada (2.10) dan (2.11).

$$g_j = \frac{\partial}{\partial \theta_j} J(\theta_j) \tag{2.10}$$

$$\theta_{j+1,} = \theta_j - \frac{\alpha}{\sqrt{\sum_{i=0}^{j} (g_i)^2}} g_j \tag{2.11}$$

Dimana:
α = initial learning rate

2.5.9 RMSProp

RMSProp (*Root Mean Square*) adalah metode pengoptimalan berbasis *adaptive learning rate* yang diusulkan oleh

Geoffrey Hinton [21]. RMSProp memodifikasi Adagrad dengan mengganti akumulasi gradien menjadi rata-rata bergerak gradien yang diberi bobot secara kuadratik, seperti yang dapat dilihat pada (2.12) dan (2.13).

$$s_j = \beta . s_{j-1} + (1-\beta)(g_j)^2 \quad (2.12)$$

$$\theta_{j+1} = \theta_j - \frac{\alpha}{\sqrt{s_j+\varepsilon}} g_j \quad (2.13)$$

Dimana:
α = initial learning rate
s_j = rata-rata eksponensial dari kuadrat gradien
g_j= gradien pada waktu j sepanjang θ_j

2.5.10 Adam

Adam (*Adaptive Moment Estimation)* juga adalah algoritma pengoptimalan yang dapat digunakan. Hampir sama dengan Adagrad, Adam memiliki sebuah *learning rate* untuk setiap parameter dan secara terpisah beradaptasi saat proses pelatihan. Adam memperbarui nilai setiap parameter seperti RMSProp [22].

$$m_j = \beta_1 . m_{j-1} - (1-\beta_1) . g_j \quad (2.14)$$

$$s_j = \beta_2 . s_{j-1} - (1-\beta_2)(g_j)^2 \quad (2.15)$$

$$\theta_{j+1} = \theta_j - \alpha \frac{m_j}{\sqrt{s_j+\varepsilon}} g_j \quad (2.16)$$

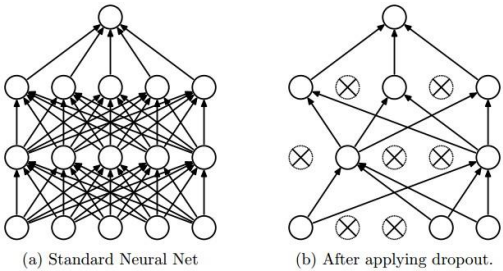
Dimana:
α = initial learning rate
g_j= gradien pada waktu j sepanjang θ_j
m_j = rata-rata eksponensial dari gradien sepanjang θ_j
s_j = rata-rata eksponensial dari kuadrat gradien sepanjang θ_j
β_1, β_2 = *hyperparameter*

Perbedaannya Adam menggunakan gradien yang telah diperhalus dan semakin mengecil seperti yang dapat dilihat pada (2.14). Lalu gradien tersebut akan digunakan untuk memperbarui parameter, seperti yang dapat dilihat pada (2.15) dan (2.16).

2.5.11 Dropout

Dropout merupakan proses mencegah terjadinya *overfitting* dan juga mempercepat proses learning. *Dropout* mengacu kepada menghilangkan neuron yang berupa *hidden layer* mapun *visible layer* di dalam jaringan [23]. Dengan menghilangkan suatu neuron, berarti menghilangkannya sementara dari jaringan yang ada. Neuron yang akan dihilangkan akan dipilih secara acak.

Pada Gambar 2.11, (a) neuron tetap utuh pada *neural network* yang belum memakai *Dropout*, dan (b) *neural network* yang sebagian dari neuronnya tidak digunakan setelah diaplikasikan Dropout.

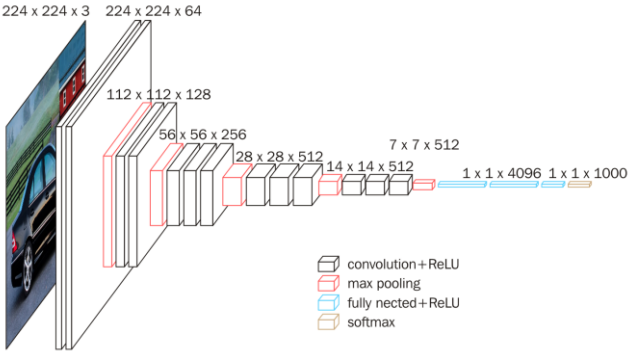


Gambar 2.11 Ilustrasi *neural network* dalam mengaplikasikan *Dropout* [24]

2.6 Single Shot Detector

Single Shot Detector (SSD) adalah sebuah *framework* terpadu untuk deteksi objek dengan satu jaringan tunggal. Arsitektur jaringan SSD dibangun berdasarkan arsitektur jaringan VGG16 yang sudah dilatih (*pre-trained*) menggunakan dataset ImageNet dengan sedikit perubahan yaitu menggunakan set *convolutional layers* tambahan untuk menggantikan *fully connected layers* pada arsitektur standar VGG16 seperti yang dapat dilihat pada Gambar 2.12. Alasan menggunakan VGG16 sebagai jaringan dasar SSD adalah hasil klasifikasi gambar yang berkualitas tinggi dan *transfer learning* yang dapat meningkatkan hasil klasifikasi [25]. Pada Tugas Akhir ini akan digunakan

arsitektur SSD berbasis VGG16. Secara utuh, arsitektur SSD dapat dilihat pada Gambar 2.13 dimana citra inputnya memiliki ukuran 300x300 *pixel* dengan 3 kanal warna (RGB). Detail arsitektur SSD dapat dilihat pada Tabel 2.1.



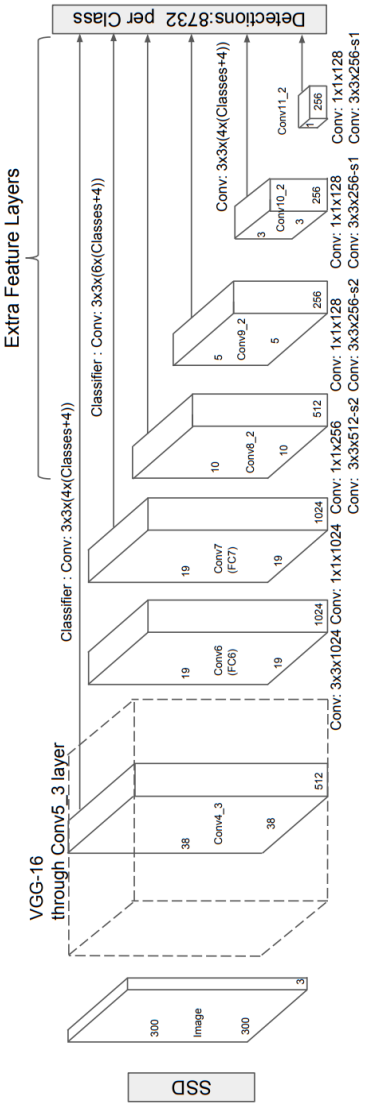
Gambar 2.12 Arsitektur VGG16 [26]

Tabel 2.1 Arsitektur SSD

Layer	**Spesifikasi**
Convolution Layer 1_1	*Filter* : 64
	Kernel : 3x3
	Stride : 1
Convolution Layer 1_2	*Filter* : 64
	Kernel : 3x3
	Stride : 1
Max Pooling Layer 1	*Kernel* : 3x3
	Stride : 2x2
Convolution Layer 2_1	*Filter* : 128
	Kernel : 3x3
	Stride : 1
Convolution Layer 2_2	*Filter* : 128
	Kernel : 3x3
	Stride : 1
Max Pooling Layer 2	*Kernel* : 3x3
	Stride : 2x2
Convolution Layer 3_1	*Filter* : 256
	Kernel : 3x3
	Stride : 1
Convolution Layer 3_2	*Filter* : 256
	Kernel : 3x3

Layer	**Spesifikasi**
	Stride : 1
Convolution Layer 3_3	*Filter* : 256
	Kernel : 3x3
	Stride : 1
Max Pooling Layer 3	*Kernel* : 3x3
	Stride : 2x2
Convolution Layer 4_1	*Filter* : 512
	Kernel : 3x3
	Stride : 1
Convolution Layer 4_2	*Filter* : 512
	Kernel : 3x3
	Stride : 1
Convolution Layer 4_3	*Filter* : 512
	Kernel : 3x3
	Stride : 1
Max Pooling Layer 4	*Kernel* : 3x3
	Stride : 2x2
Convolution Layer 5_1	*Filter* : 512
	Kernel : 3x3
	Stride : 1
Convolution Layer 5_2	*Filter* : 512
	Kernel : 3x3
	Stride : 1
Convolution Layer 5_3	*Filter* : 512
	Kernel : 3x3
	Stride : 1
Max Pooling Layer 5	*Kernel* : 3x3
	Stride : 1x1
Convolution Layer 6	*Filter* : 1024
	Kernel : 3x3
	Stride : 1
	Padding : SAME
Convolution Layer 7	*Filter* : 1024
	Kernel : 1x1
	Stride : 1
	Padding : SAME
Convolution Layer 8_1	*Filter* : 256
	Kernel : 1x1
	Stride : 1
	Padding : SAME
Convolution Layer 8_2	*Filter* : 512
	Kernel : 3x3
	Stride : 2
	Padding : SAME
Convolution Layer 9_1	*Filter* : 128

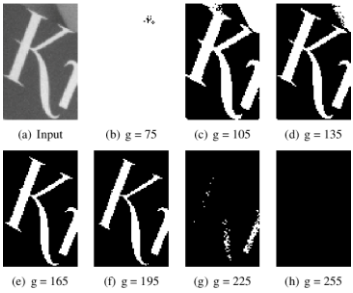
Layer	**Spesifikasi**
	Kernel : 1x1
	Stride : 1
	Padding : SAME
Convolution Layer 9_2	*Filter* : 256
	Kernel : 3x3
	Stride : 2
	Padding : SAME
Convolution Layer 10_1	*Filter* : 128
	Kernel : 1x1
	Stride : 1
	Padding : SAME
Convolution Layer 10_2	*Filter* : 256
	Kernel : 3x3
	Stride : 1
	Padding : VALID
Convolution Layer 11_1	*Filter* : 128
	Kernel : 1x1
	Stride : 1
	Padding : SAME
Convolution Layer 11_2	*Filter* : 256
	Kernel : 3x3
	Stride : 1
	Padding : VALID



Gambar 2.13 Arsitektur SSD [25]

2.7 Maximally Stable Extremal Regions

Maximally Stable Extremal Regions (MSER) adalah metode untuk mendeteksi *blob* dalam sebuah citra. MSER banyak digunakan pada aplikasi deteksi dan pengenalan teks. MSER didasarkan pada ide dalam mengambil daerah yang hampir sama melalui berbagai nilai ambang (*threshold*) [27]. Dasar perhitungan MSER dimulai dengan mengurutkan *pixel* pada gambar dari intensitas rendah ke yang lebih tinggi atau sebaliknya (misal pada citra *grayscale* yang memiliki intensitas {0, ... ,255}). Intensitas ini yang dinamakan *threshold*. Iterasi dimulai dari *threshold* rendah (0) ke yang lebih tinggi (255) dan pada masing-masing *threshold* dilakukan perhitungan area, dimana area yang tidak mengalami perubahan ketika *threshold* diubah-ubah dinamakan MSER *regions*.



Gambar 2.14 Proses analisis citra *threshold* menggunakan MSER [28]

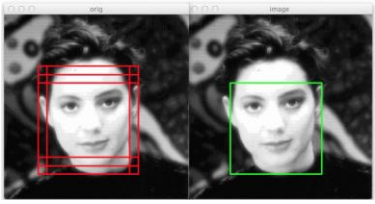
Dalam setiap *threshold* citra, area ekstremal ditandai sebagai komponen yang berhubungan, sehingga terbentuk sebuah urutan dari masing-masing komponen. Pada beberapa *threshold*, dua atau lebih komponen akan bergabung menjadi satu. Pada *threshold* ini terjadi perubahan bentuk komponen secara signifikan, hal ini menjadikan komponen tidak stabil dan letak lokasi menjadi tidak presisi, terutama dalam hal perubahan intensitas dan *noise*. MSER *regions* akan diekstrak dari urutan komponen pada *threshold* yang direpresentasikan oleh perbedaan minimum komponen lokal dalam area di dalam ruang *threshold* [29]. Pada Gambar 2.14 dapat dilihat bagian (a) menunjukkan area yang dipertimbangkan dan bagian (b)

sampai (g) menunjukkan hasil citra *threshold* pada tingkat abu-abu, g. Hasilnya, huruf K diidentifikasi sebagai MSER *regions* karena ukuran area yang terhubung tidak berubah secara signifikan dalam kisaran tingkat abu-abu dari 135-195 [28]. Pada Tugas Akhir ini metode MSER akan digunakan untuk segmentasi karakter pada pelat nomor kendaraan.

2.8 Non-Maximum Suppression

Saat menggunakan metode *Maximally Stable Extremal Regions* (MSER) untuk deteksi objek biasanya akan didapatkan beberapa *bounding box* yang tumpang tindih untuk setiap objek. Untuk menghilangkan *bounding box* yang saling tumpang tindih tersebut sehingga hanya ada satu *bounding box* untuk setiap objek yang dideteksi akan digunakan *Non-Maximum Suppression* (NMS), sebuah metode *post-processing* populer yang dapat digunakan untuk menghilangkan *bounding box* yang *redundant*.

Setelah menerapkan metode MSER untuk prediksi *bounding box* yang sudah dijelaskan di atas, maka akan didapatkan serangkaian *bounding box* B dengan dengan koordinat (x_1, y_1) dan (x_2, y_2) untuk kategori objek tertentu dalam citra. Setelah itu *bounding box* B diurutkan berdasarkan koordinat titik y_2 dan akan dipilih *bounding box* dengan y_2 terbesar dan menghapus *bounding box* yang *overlap* dengan *bounding box* dari seleksi sebelumnya sebesar nilai *threshold* yang ditentukan [30] [31].



Gambar 2.15 Eliminasi *bounding box* pada citra sebelum (kiri) dan sesudah (kanan) menggunakan NMS [31]

2.9 Akurasi, Presisi, Recall, dan F1-Score

Evaluasi terhadap kinerja suatu algoritma klasifikasi dilakukan untuk mengetahui seberapa baik algoritma dalam mengklasifikasikan data. *Confusion matrix* merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu algoritma klasifikasi. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh algoritma dengan hasil klasifikasi yang seharusnya [32].

Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat empat istilah sebagai representasi hasil proses klasifikasi yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN). Nilai TN merupakan jumlah data negatif yang terdeteksi dengan benar, sedangkan FP merupakan data negatif namun terdeteksi sebagai data positif. Sementara TP merupakan data positif yang terdeteksi benar dan FN merupakan kebalikan dari TP, yaitu data positif namun terdeteksi sebagai data negatif. Pada jenis klasifikasi biner yang hanya memiliki dua kelas, *confusion matrix* disajikan seperti pada Tabel 2.2.

Tabel 2.2 *Confusion matrix*

		Kelas Prediksi	
		Benar	Salah
Kelas Sebenarnya	Benar	*True Positive* (TP)	*False Negative* (FN)
	Salah	*False Positive* (FP)	*True Negative* (TN)

Berdasarkan nilai TN, FP, FN, dan TP dapat diperoleh nilai akurasi, presisi, *recall*, dan *f1-score*. Nilai akurasi pada (2.17) menggambarkan seberapa akurat algoritma dapat mengklasifikasikan data secara benar, dengan kata lain, nilai akurasi merupakan perbandingan antara data yang terklasifikasi benar dengan keseluruhan data. Nilai presisi pada (2.18) menggambarkan jumlah data kategori positif yang diklasifikasikan secara benar dibagi dengan total data yang diklasifikasi positif.

Nilai *recall* pada (2.19) menunjukkan berapa persen data kategori positif yang terklasifikasikan dengan benar. Sementara, nilai *f1-score* pada (2.20) menggambarkan rata-rata harmonis antara nilai presisi dan nilai *recall.*

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.17}$$

$$Presisi = \frac{TP}{TP + FP} \tag{2.18}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.19}$$

$$F1 - score = \frac{2\ x\ Presisi\ x\ Recall}{Presisi + Recall} \tag{2.20}$$

Dimana:

1. TP adalah *True Positive*, yaitu jumlah data positif yang terklasifikasi dengan benar.
2. TN adalah *True Negative*, yaitu jumlah data negatif yang terklasifikasi dengan benar.
3. FN adalah *False Negative*, yaitu jumlah data negatif namun terklasifikasi salah.
4. FP adalah *False Positive*, yaitu jumlah data positif namun terklasifikasi salah.

Sementara itu, pada klasifikasi dengan jumlah kelas yang lebih dari dua atau multikelas, cara menghitung akurasi, presisi dan *recall* dapat dilakukan dengan menghitung rata-rata dari nilai akurasi, presisi dan recall pada setiap kelas [33] seperti yang dapat dilihat pada (2.21), (2.22), dan (2.23), sedangkan *f f1-score* perhitungannya sama dengan klasifikasi dengan jumlah kelas sebanyak dua.

$$Akurasi = \frac{\sum_{i=1}^{l} \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}}{l} \tag{2.21}$$

$$Presisi = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l}(TP_i + FP_i)} \quad (2.22)$$

$$Recall = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l}(TP_i + FN_i)} \quad (2.23)$$

Dimana:

1. TP_i adalah *True Positive*, yaitu jumlah data positif yang terklasifikasi dengan benar untuk kelas ke-i.
2. TN_i adalah *True Negative*, yaitu jumlah data negatif yang terklasifikasi dengan benar untuk kelas ke-i.
3. FN_i adalah *False Negative*, yaitu jumlah data negatif namun terklasifikasi salah untuk kelas ke-i.
4. FP_i adalah *False Positive*, yaitu jumlah data positif namun terklasifikasi salah untuk kelas ke-i
5. l adalah jumlah kelas.

Perhitungan perhitungan akurasi, presisi, *recall* dan *f1-score* untuk multikelas akan dimanfaatkan sebagai evaluasi pada saat klasifikasi karakter pada pelat.

2.10 Python

Python adalah bahasa pemograman yang populer. Python sering dimanfaatkan dalam pengembangan web, perangkat lunak, penelitian, dan *system scripting*. Python dapat digunakan untuk menangani data besar dan melakukan operasi matematika yang kompleks. Python bekerja di berbagai *platform* seperti Windows, Mac, Linux, Raspberry Pi, dan lain-lain. Python dirancang untuk mudah dibaca, yaitu memiliki sintaks yang sederhana dan menggunakan bahasa inggris [34].

2.11 Library

Library merupakan sekumpulan program yang dapat digunakan pada program lain tanpa terikat satu dengan yang lainnya. Terdapat beberapa *library* yang digunakan dalam melakukan implementasi tugas akhir ini. *Library* yang digunakan

antara lain, Keras, TensorFlow, OpenCV, Python Imaging Library (PIL), Numpy, Scikit-learn, dan Matplotlib.

2.11.1 Keras

Keras adalah *high-level neural networks* API, yang ditulis dalam bahasa pemograman Python dan mampu berjalan di atas TensorFlow dan Theano. Keras dapat berjalan baik di CPU dan GPU. Keras berisi banyak implementasi *neural network* yang umum digunakan, fungsi aktivasi, *optimizer*, dan *tool* lain yang memudahkan dalam pengolahan citra dan data teks [35].

2.11.2 TensorFlow

TensorFlow merupakan *open source library* yang dimanfaatka untuk untuk pembuatan program yang membutuhkan komputasi numerik berkinerja tinggi. TensorFlow dikembangakan oleh tim Google Brain. TensorFlow menyediakan fungsi-fungsi *machine learning* dan *deep learning*, dan dapat dijalankan dalam CPU atau GPU [36].

2.11.3 OpenCV

OpenCV (*Open Source Computer Vision*) adalah *library* yang dimanfaatkan dalam pengolahan citra dinamis secara *real-time*. OpenCV dapat digunakan dalam berbagai bahasa pemograman seperti Python, C++, Java, atau MATLAB. OpenCV memiliki fitur seperti *Feature & Object Detection, Motion Analysis and Object Tracking, Image Filtering, Image Processing*, dan lain-lain [37].

2.11.4 Python Imaging Library (PIL)

Python Imaging Library (PIL) merupakan *open source library* untuk bahasa pemrograman Python yang dimanfaatkan untuk membuka, memanipulasi, dan menyimpan gambar dalam banyak format yang berbeda [38].

2.11.5 Numpy

Numpy adalah *library Python* yang mendukung pengolahan data pada *array* dan matriks multidimensi yang besar. Numpy menyediakan kumpulan fungsi matematika, seperti aljabar linear, transformasi Fourier, pembuatan angka acak, dan lain-lain. Numpy bersifat *open source* sehingga banyak dimanfaatkan dalam pengolahan data peneilitian [39].

2.11.6 Scikit-learn

Scikit-learn merupakan *open source machine learning library* untuk bahasa pemograman Python. Scikit-learn menyediakan fitur seperti *classification*, *regression*, *clustering*, termasuk juga didalamnya algoritma *Support Vector Machines* (SVM), *random forest*, *gradient boosting*, dan lain-lain [40].

2.11.7 Matplotlib

Matplotlib adalah *library* Python yang mendukung pembuatan grafik dua dimensi dalam berbagai format dan dari berbagai jenis data. Matplotlib bersifat *open source* dan banyak digunakan untuk pengolahan data dalam penelitian. Matplotlib dapat membuat plot, histogram, spektrum daya, diagram batang, diagram kesalahan, plot pencar, dan lain-lain [41].

2.12 Node.js

Node.js merupakan salah satu *platform* pengembang yang dapat digunakan untuk membuat aplikasi berbasis *cloud*. Node.js dikembangkan dari *engine* JavaScript yang dibuat oleh *Google* untuk *browser Chrome* ditambah dengan libUV serta beberapa pustaka lainnya. Node.js menggunakan JavaScript sebagai bahasa pemrograman dan *event-driven*, *non-blocking* I/O (*asynchronous*) model yang membuatnya ringan dan efisien. Node.js memiliki fitur *built-in HyperText Transfer Protocol* (HTTP) *server library* yang menjadikannya mampu menjadi sebuah *web server* tanpa bantuan *software* lainnya seperti Apache dan Nginx [42].

2.13 Express.js

Express.js adalah sebuah kerangka kerja aplikasi berbasis *web* yang menggunakan *core* pemrograman Node.js dengan komponen modul HTTP dan Connect [43]. Kerangka kerja ini dibuat untuk berjalan dalam mesin V8 Chrome, membuatnya berjalan beriringan dengan Node.js sehingga dapat melakukan kerja dengan sangat cepat.

2.14 Socket.io

Socket.io merupakan Javascript *library* untuk membuat *website* yang *real-time*, dan memungkinkan komunikasi dua arah secara *real-time* antara *web client* dan *server*. Socket.io memiliki dua bagian *library* untuk *client-side* yang berjalan di *browser* dan *server-side library* untuk Node.js [44].

2.15 Redis

Redis adalah struktur data yang dapat digunakan sebagai database, *cache* dan penghubung pesan [45]. Redis sangat cocok untuk mendukung model *Publish*/*Subscribe* dimana aplikasi *publisher* mengirim pesan ke satu atau lebih *channel* yang dimonitor oleh aplikasi *subscriber* yang akan merespon pesan tersebut [46].

2.16 *HyperText Markup Language* (HTML)

HyperText Markup Language (HTML) adalah bahasa standar pemrograman untuk membuat halaman *web* yang terdiri dari kode-kode *tag* tertentu, kemudian kode-kode tersebut diterjemahkan oleh *web browser* untuk menampilkan halaman web yang terdiri dari beberapa macam format tampilan seperti teks, grafik, animasi *link*, maupun audio-video [47].

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan tentang perancangan data dan sistem pengenalan pelat nomor kendaraan menggunakan *Convolutional Neural Network*. Bab ini juga akan menjelaskan gambaran umum sistem dalam bentuk diagram alir.

3.1 Perancangan Data

Dalam Tugas Akhir ini, dataset yang akan digunakan sebagai data latih berupa citra pelat dan karakter sintetis yang dibuat mandiri dengan spesifikasi pelat nomor kendaraan standar Indonesia. Spesifikasi dataset citra pelat sintetis dapat dilihat pada Tabel 3.1 dan spesifikasi dataset citra karakter sintetis dapat dilihat pada Tabel 3.2.

Tabel 3.1 Spesifikasi dataset citra pelat sintetis

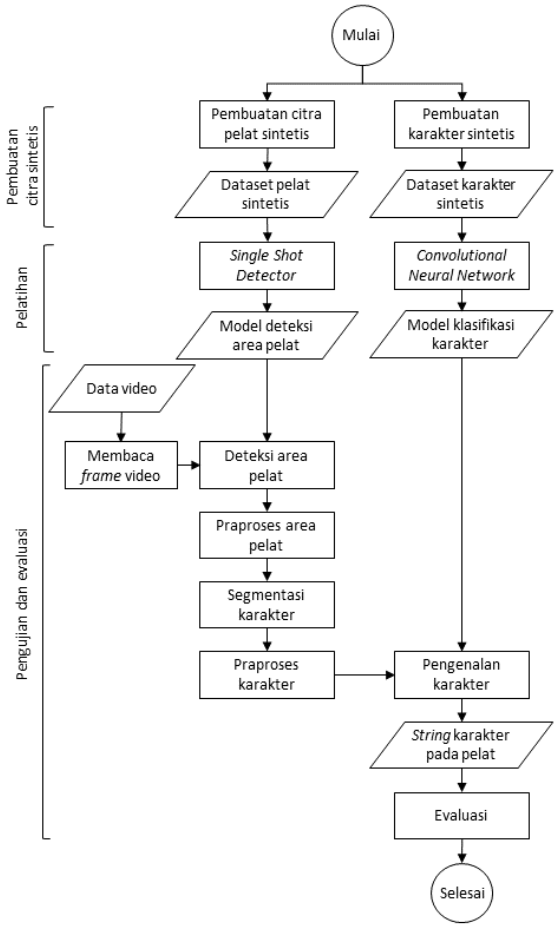
Keterangan	Spesifikasi
Ukuran gambar	533x300 *pixels*
Jumlah gambar	20.000
Ukuran file	17-870 KB
Ekstensi	.png
Kanal warna	3 (RGB)

Tabel 3.2 Spesifikasi dataset citra karakter sintetis

Keterangan	Spesifikasi
Ukuran gambar	24x40 *pixels*
Jumlah gambar	54.000
Jumlah kelas	36
Jumlah gambar per kelas	1.500
Ukuran file	1-2 KB
Ekstensi	.jpg
Kanal warna	3 (RGB)

3.2 Desain Umum Sistem

Sistem pengenalan pelat nomor kendaraan yang dibangun memiliki tiga proses utama yaitu pembuatan citra sintetis, pelatihan, pengujian dan evaluasi *Single Shot Detector* dan *Convolutional Neural Network*. Diagram alir dari sistem ditunjukan pada Gambar 3.1.



Gambar 3.1 Diagram alir sistem yang dibangun

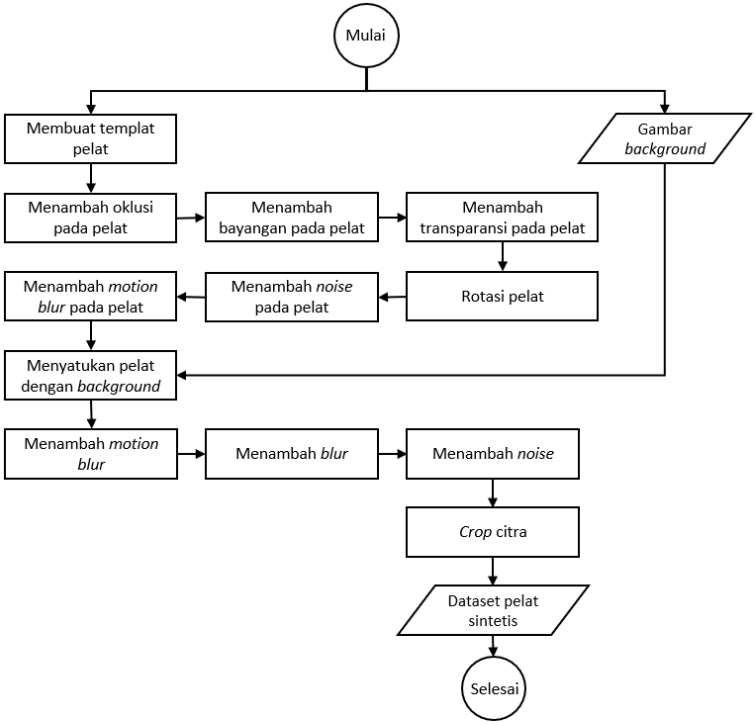
Akan dilakukan pembuatan citra pelat dan karakter sintetis yang masing-masing digunakan sebagai data latih untuk deteksi area pelat menggunakan *Single Shot Detector* (SSD) dan pengenalan karakter pada pelat menggunakan *Convolutional Neural Network* (CNN). Pada proses pelatihan, data latih yang telah dibuat akan diekstraksi fiturnya melalui proses konvolusi, *pooling*, dan fungsi-fungsi aktivasi. Hasil dari proses pelatihan tersebut akan menjadi model untuk proses deteksi area pelat dan klasifikasi karakter pada pelat. Pada proses pengujian, akan dilakukan praproses dengan cara konversi warna ke *grayscale* pada citra pelat yang berhasil dideteksi sebelum masuk ke proses segmentasi karakter. Karakter pada pelat akan disegmentasi menggunakan metode *Maximally Stable Extremal Regions*. Selanjutnya, akan dilakukan praproses pada citra karakter yang tersegmentasi sebelum dilakukan prediksi kelas. Proses pengujian memanfaatkan fungsi *softmax* untuk memprediksi label kelas dengan menghitung nilai probabilitas tertinggi. Selanjutnya hasil prediksi label kelas tersebut akan dibandingkan dengan label kelas sebenarnya dan model dievaluasi dengan menghitung nilai akurasi, *precision*, *recall*, dan *f1-score*-nya.

3.2.1 Tahap Pembuatan Dataset Pelat Sintetis

Dataset yang digunakan sebagai data uji untuk deteksi pelat nomor kendaraan berupa citra pelat sintetis. Alir pembuatan citra pelat sintetis dapat dilihat pada Gambar 3.2. Gambar *background* yang digunakan dalam pembuatan citra pelat sintetis diambil dari MIRFLICKR-1M dataset [48] yang terdiri dari 1 juta gambar dengan variasi ukuran dan 862.115 *tag* yang berbeda diambil dari situs fotografi Flickr. Contoh gambar dari MIRFLICKR-1M dataset dapat dilihat pada Gambar 3.3 dan frekuensi *tag* yang paling banyak pada dataset ini dapat dilihat pada Gambar 3.4.

Pada Tugas Akhir ini hanya akan digunakan sebanyak 20.000 gambar sebagai *background* pada proses pembuatan citra pelat sintetis. Gambar yang digunakan akan diseragamkan ukurannya dengan meng-*crop* gambar ke dalam ukuran 768x384

pixels. Contoh gambar dari MIRFLICKR-1M dataset setelah di-*crop* dapat dilihat pada Gambar 3.5.



Gambar 3.2 Diagram alir pembuatan citra pelat sintetis

Gambar 3.3 Contoh gambar pada MIRFLICKR-1M dataset

Tag	# Images	Tag	# Images
sky	845	people	330
water	641	city/urban	308/247
portrait	623	sea	301
night	621	sun	290
nature	596	girl	262
sunset	585	snow	256
clouds	558	food	225
flower/flowers	510/351	bird	218
beach	407	sign	214
landscape	385	car	212
street	383	lake	199
dog	372	building	188
architecture	354	river	175
graffiti/streetart	335/184	baby	167
tree/trees	331/245	animal	164

Gambar 3.4 Frekuensi tag pada MIRFLICKR-1M dataset [49]

Gambar 3.5 Contoh gambar dari MIRFLICKR-1M dataset setelah di-*crop*

3.2.1.1 Tahap Pembuatan Templat Pelat

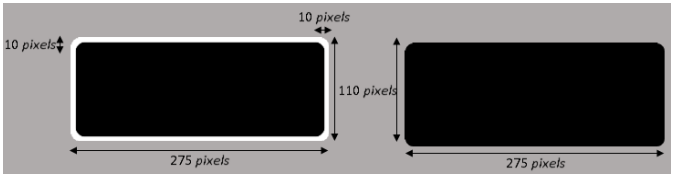
Pada tahap ini akan dibuat templat pelat dengan alir yang dapat dilihat pada Gambar 3.6. Spesifikasi templat pelat nomor kendaraan bermotor standar Indonesia [50] yang akan dibuat dapat dilihat pada Tabel 3.3. Akan ada dua jenis dasar pelat yang berbeda yaitu dengan pinggiran dan tanpa pinggiran yang dibuat dengan probabilitas masing-masing sebesar 50% seperti yang dapat dilihat pada Gambar 3.7. Tahap berikutnya adalah penambahan tulisan pada dasar pelat yang terdiri dari enam bagian dengan spesifikasi yang dapat dilihat pada Tabel 3.4. Terdapat tujuh *font* berbeda yang digunakan untuk menambah variasi gaya tulisan yaitu Indonesia

License Plate [51], Arial Bold, Arial Narrow, MS Gothic Regular, Ebrima, Consolas, dan Segoe MDL2 Assets. Beberapa contoh *font* yang digunakan dapat dilihat pada Gambar 3.8.

Gambar 3.6 Diagram alir pembuatan templat pelat

Tabel 3.3 Spesifikasi pelat nomor standar Indonesia

Keterangan	Spesifikasi
Ukuran pelat	275x110 *pixels*
Warna dasar pelat	Hitam
Warna pinggiran pelat (optional)	Putih
Warna tulisan	Putih



Gambar 3.7 Dua jenis dasar plat dan spesifikasi ukurannya. Atas: dengan pinggiran. Bawah: tanpa pinggiran

Gambar 3.8 Beberapa jenis *font* yang digunakan. (a) Indonesia License Plate; (b) Arial Bold; (c) Arial Narrow; (d) MS Gothic Regular; (e) Ebrima.

Tabel 3.4 Spesifikasi tulisan pada templat pelat

Bagian	**Spesifikasi**	**Gambar**	**Ukuran (*pixels*)**
Kode wilayah	Terdiri dari 1-2 huruf dan ditempatkan pada baris pertama pelat		26x44
			52x44
Nomor polisi	Terdiri dari 4 angka dan ditempatkan pada baris pertama setelah kode wilayah		102x44
Kode akhir wilayah	Terdiri dari 1-2 huruf dan ditempatkan pada baris pertama setelah nomor polisi		26x44
			52x44
Bulan masa berlaku	Terdiri dari 2 angka dan ditempatkan pada baris kedua pelat		30x26
Dot	Ditempatkan pada baris kedua pelat setelah bulan masa berlaku		15x41
Tahun masa berlaku	Terdiri dari 2 angka dan ditempatkan pada baris kedua pelat setelah dot		31x26

Setiap bagian tulisan (kecuali dot) akan di-*random* karakter dan jenis *font*-nya untuk setiap pelat. Contoh hasil dari dua jenis templat pelat setelah penambahan tulisan dapat dilihat pada Gambar 3.9.

Gambar 3.9 Hasil penyatuan 2 jenis templat pelat dengan tulisan. Kiri: dengan pinggiran. Kanan: tanpa pinggiran

Untuk menambah variasi warna pada pelat dipilih secara *random* warna hitam dan putih [52] dengan batasan sebagai berikut:

Hitam = (K_r, K_g, K_b)

dimana $K_r, K_g, K_b \in \mathbb{N}_0[0, 60]$

Putih = $(L+W_r, L+W_g, L+W_b)$

dimana $L \in \mathbb{N}_0$ [155, 225] dan

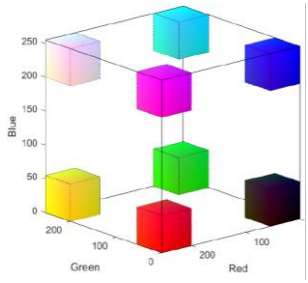
$W_r, W_g, W_b \in \mathbb{N}_0$ [0, 30]

Contoh hasil pelat setelah diberikan variasi warna hitam dan putih dapat dilihat pada Gambar 3.10.

Gambar 3.10 Hasil pelat dengan variasi warna.
Atas: dengan pinggiran. Bawah: tanpa pinggiran

3.2.1.2 Tahap Penambahan Oklusi pada Pelat

Pada kenyataannya terkadang bagian pelat nomor tertutup atau terhalang sebagian oleh komponen kendaraan. Untuk menyimulasikan kondisi seperti itu, dilakukan penambahan oklusi pada pelat dengan probabilitas 50%. Oklusi ditambah dengan warna yang mewakili warna kendaraan dan ukuran 1-15% dari lebar pelat (untuk oklusi horizontal) atau panjang pelat (untuk oklusi vertikal). Warna oklusi dipilih secara random dari R, G, B $\in \mathbb{N}_0[0, 55] \cup [200, 255]$ [52] yang diilustrasikan pada Gambar 3.11.



Gambar 3.11 Rentang warna oklusi [52]

Oklusi secara random diletakkan pada salah satu bagian pelat (bagian atas dengan probabilitas 50%, sebelah kiri dengan probabilitas 25%, atau sebelah kanan dengan probabilitas 25%). Contoh hasil pelat setelah ditambah oklusi dapat dilihat pada Gambar 3.12.

Gambar 3.12 Hasil pelat dengan oklusi. Atas: dengan pinggiran. Bawah: tanpa pinggiran

3.2.1.3 Tahap Penambahan Bayangan pada Pelat

Pada kenyataannya terkadang sinar dari matahari sumber lainnya menyebabkan adanya bayangan pada bagian pelat nomor. Untuk menyimulasikan kondisi seperti itu, dilakukan penambahan bayangan atau *shadow* pada pelat dengan probabilitas 50%. Bayangan ditambah dengan menggelapkan warna pelat sebesar 10-70% dengan ukuran 10-70% dari lebar pelat (untuk bayangan horizontal) atau panjang pelat (untuk bayangan vertikal) [52]. Bayangan secara random diletakkan pada salah satu bagian pelat (bagian atas dengan probabilitas 50%, sebelah kiri dengan probabilitas 25%, atau sebelah kanan dengan probabilitas 25%).

Contoh hasil pelat setelah ditambah bayangan dapat dilihat pada Gambar 3.13.

Gambar 3.13 Hasil pelat dengan bayangan. Atas: dengan pinggiran. Bawah: tanpa pinggiran

3.2.1.4 Tahap Penambahan Transparansi pada Pelat

Untuk menyimulasikan kondisi lain dimana pelat terhalang oleh objek-objek lain, dilakukan penambahan transparansi pada pelat dengan probabilitas 50%. Transparansi ditambah dengan ukuran 1-15% dari lebar pelat (untuk bayangan horizontal) atau panjang pelat (untuk bayangan vertikal) [52]. Transparansi secara *random* diletakkan pada salah satu bagian pelat (bagian atas dengan probabilitas 25%, bagian bawah dengan probabilitas 25%, sebelah kiri dengan probabilitas 25%, atau sebelah kanan dengan probabilitas 25%). Contoh hasil pelat setelah ditambah transparansi dapat dilihat pada Gambar 3.14.

Gambar 3.14 Hasil pelat dengan transparansi.
Atas: dengan pinggiran. Bawah: tanpa pinggiran

3.2.1.5 Tahap Rotasi Pelat

Untuk menyimulasikan kondisi nyata dimana citra pelat biasanya diambil dari sudut dan jarak pengambilan yang berbeda-beda, dilakukan rotasi pada pelat seperti yang sudah dijelaskan pada subbab 2.3.3. Pelat dirotasi pada sumbu x sebesar θ^o, sumbu

y sebesar ϕ^o, dan sumbu z sebesar ψ^o dengan batasan sebagai berikut:

1. θ, dimana $-60° \leq \theta \leq 60°$
2. ϕ, dimana $-60° \leq \phi \leq 70°$
3. Total sudut $|\theta| + |\phi| < 90°$
4. ψ, dimana $-5° \leq \psi \leq 5°$

Batasan sudut diatas didapatkan dari [52] dengan beberapa penyesuaian batasan pada sudut θ dan ϕ karena adanya perbedaan dimensi pelat Indonesia dan pelat pada [52]. Contoh hasil pelat setelah dirotasi dengan batasan sudut diatas dapat dilihat pada Tabel 3.5.

Tabel 3.5 Hasil pelat setelah dirotasi

Rotasi	Sudut Min	Gambar	Sudut Maks	Gambar
θ	$-60°$		$60°$	
ϕ	$-60°$		$70°$	
ψ	$-5°$		$5°$	
$\|\theta\|+\|\phi\|$	-	-	$90°$	

3.2.1.6 Tahap Penambahan Noise pada Pelat

Untuk menyimulasikan kondisi dimana pada permukaan pelat terdapat kotoran atau debu, dilakukan penambahan *noise* pada pelat. *Noise* ditambah dengan mengaplikasikan *Gaussian noise* yang sudah dijelaskan pada subbab 2.3.1 dengan *mean* = 0 dan standar deviasi berada dalam rentang 0-10 (dipilih secara

random untuk tiap pelat). Contoh hasil pelat setelah ditambah *noise* dapat dilihat pada Gambar 3.15.

Gambar 3.15 Hasil pelat dengan *noise*. Atas: dengan pinggiran. Bawah: tanpa pinggiran

3.2.1.7 Tahap Penambahan Motion Blur pada Pelat

Untuk menyimulasikan kondisi dimana pelat terkadang diambil dalam keadaan kendaraan bergerak, dilakukan penambahan *motion blur* pada pelat. *Motion blur* ditambah dengan mengaplikasikan *filter motion blur* secara horizontal atau vertikal dengan probabilitas masing-masing 50%. Contoh hasil pelat setelah ditambah *motion blur* dapat dilihat pada Gambar 3.16.

Gambar 3.16 Hasil pelat dengan *motion blur*.
Atas: dengan pinggiran (*motion blur* horizontal). Bawah: tanpa pinggiran (*motion blur* vertikal)

3.2.1.8 Tahap Penyatuan Pelat dengan Background

Pada tahap ini citra pelat sintetis yang sudah dibuat akan disatukan dengan *background*. Citra pelat akan diletakkan pada bagian tengah *background*. Contoh hasil pelat setelah disatukan dengan *background* dapat dilihat pada Gambar 3.17.

Gambar 3.17 Hasil penyatuan pelat dengan *background*

3.2.1.9 Tahap Penambahan Motion Blur

Pada tahap ini akan ditambahkan *motion blur* pada citra pelat yang sudah disatukan dengan *background* dengan probabilitas 50%. *Motion blur* ditambah dengan mengaplikasikan *filter motion blur* secara horizontal atau vertikal dengan probabilitas masing-masing 50%. Contoh hasil citra pelat dan *background* setelah ditambah *motion blur* dapat dilihat pada Gambar 3.18.

Gambar 3.18 Hasil citra pelat dan *background* dengan *motion blur*. Atas:(*motion blur* horizontal. Bawah: *motion blur* vertikal

3.2.1.10 Tahap Penambahan Blur

Pada tahap ini akan ditambahkan *blur* pada citra pelat yang sudah disatukan dengan *background* dengan probabilitas 50% untuk menyimulasikan kondisi fokus pengambilan gambar yang kabur. Untuk menambah *blur Gaussian filter* diaplikasikan dengan radius sama dengan 1. Contoh hasil citra pelat dan *background* setelah ditambah *blur* dapat dilihat pada Gambar 3.19.

Gambar 3.19 Hasil citra pelat dan *background* dengan *blur*

3.2.1.11 Tahap Penambahan Noise

Untuk menyimulasikan kondisi dimana kualitas gambar yang diambil kurang bagus, dilakukan penambahan *noise* pada citra pelat yang sudah disatukan dengan *background* dengan probabilitas 50%. *Noise* ditambah dengan mengaplikasikan *Gaussian noise* yang sudah dijelaskan pada subbab 2.3.1 dengan *mean* = 0 dan standar deviasi berada dalam rentang 0-8 (dipilih secara *random* untuk tiap citra). Contoh hasil citra pelat dan *background* setelah ditambah *noise* dapat dilihat pada Gambar 3.20.

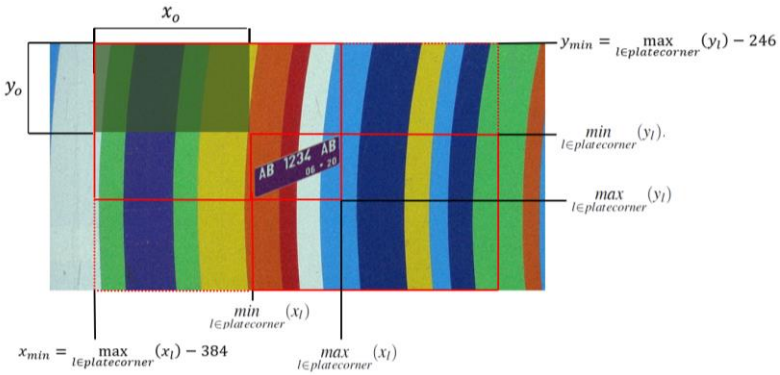
Gambar 3.20 Hasil citra pelat dan *background* dengan *noise*

3.2.1.12 Tahap Crop Citra

Sampel pelat untuk data latih didapatkan dengan melakukan *crop* dengan ukuran 533x300 *pixels* pada citra pelat dan *background* yang sudah dibuat sebelumnya. Lokasi *crop* (x_o, y_o) untuk setiap citra akan dipilih *random* pada rentang [x_{min}, x_{max}] dan [y_{min}, y_{max}] dengan batasan pada (3.1) sehingga seluruh bagian pelat akan ada di dalam citra setelah proses *crop* seperti yang dapat

dilihat pada Gambar 3.21. Contoh hasil citra yang telah di-*crop* dapat dilihat pada Gambar 3.22.

$$
\begin{aligned}
x_{min} &= \max_{l \in platecorner} (x_l) - 384 \\
x_{max} &= \min_{l \in platecorner} (x_l) \\
y_{min} &= \max_{l \in platecorner} (y_l) - 246 \\
y_{max} &= \min_{l \in platecorner} (y_l)
\end{aligned}
\tag{3.1}
$$



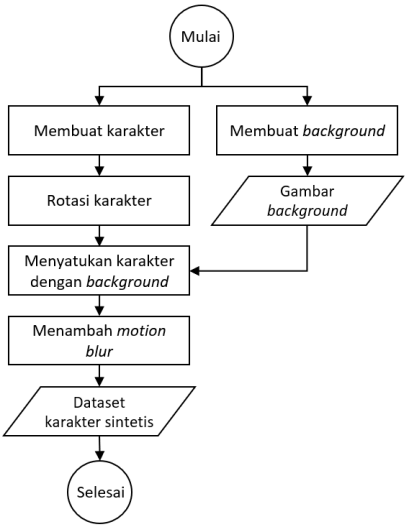
Gambar 3.21 Ilustrasi area *crop* citra

Gambar 3.22 Hasil citra setelah di-*crop*

3.2.2 Tahap Pembuatan Dataset Karakter Sintetis

Dataset yang digunakan sebagai data uji untuk klasifikasi karakter pada pelat nomor kendaraan berupa citra karakter sintetis.

Alir pembuatan citra karakter sintetis dapat dilihat pada Gambar 3.23.



Gambar 3.23 Diagram alir pembuatan citra karakter sintetis

3.2.2.1 Tahap Pembuatan Karakter

Pada tahap ini akan dibuat sebanyak 36 jenis citra karakter yang terdiri dari 26 karakter huruf (A-Z) dan 10 karakter angka (0-9). Setiap citra akan di-*random* jenis *font* dan warnanya. Untuk variasi warna pada karakter dipilih secara *random* putih [52] dengan batasan sebagai berikut:

$$\text{Putih} = (L+W_r, L+W_g, L+W_b)$$

dimana $L \in \mathbb{N}_0$ [155, 225] dan

$W_r, W_g, W_b \in \mathbb{N}_0$ [0, 30]

Contoh hasil citra karakter yang dibuat dapat dilihat pada Gambar 3.24.

Gambar 3.24 Hasil citra karakter yang dibuat

3.2.2.2 Tahap Rotasi Karakter

Untuk menyimulasikan kondisi nyata dimana karakter pada pelat biasanya memiliki sudut dan jarak pengambilan yang berbeda-beda, dilakukan rotasi pada karakter seperti yang sudah dijelaskan pada subbab 2.3.3. Karakter dirotasi pada sumbu x sebesar θ^o, sumbu y sebesar ϕ^o, dan sumbu z sebesar ψ^o dengan batasan sebagai berikut:

1. θ, dimana $-40° \leq \theta \leq 40°$
2. ϕ, dimana $-40° \leq \phi \leq 40°$
3. ψ, dimana $-5° \leq \psi \leq 5°$

Contoh hasil karakter setelah dirotasi dengan batasan sudut diatas dapat dilihat pada Tabel 3.5.

Tabel 3.6 Hasil karakter setelah dirotasi

Rotasi	Sudut Min	Gambar	Sudut Maks	Gambar
θ	$-40°$		$40°$	
ϕ	$-40°$		$40°$	
ψ	$-5°$		$5°$	

3.2.2.3 Tahap Pembuatan Background

Setelah proses rotasi, biasanya citra karakter akan memiliki dimensi (panjang dan lebar) yang berbeda-beda, sehingga perlu dibuat *background* sebagai dasar karakter untuk menyamakan ukuran citra karakter sintetis. *Background* dibuat dengan ukuran 24x40 *pixels* dan warna hitam yang dipilih secara *random* untuk tiap karakter dengan batasan sebagai berikut:

Hitam $= (K_r, K_g, K_b)$

$$\text{dimana } K_r, K_g, K_b \in \mathbb{N}_0[0, 60]$$

Contoh *background* dapat dilihat Gambar 3.25.

Gambar 3.25 *Background* untuk karakter dengan variasi warna hitam

3.2.2.4 Tahap Penyatuan Karakter dengan Background

Pada tahap ini citra karakter sintetis akan disatukan dengan *background* yang sudah dibuat. Citra karakter akan diletakkan pada bagian tengah *background.* Contoh hasil karakter setelah disatukan dengan *background* dapat dilihat Gambar 3.26.

Gambar 3.26 Hasil penyatuan karakter dengan *background*

3.2.2.5 Tahap Penambahan Motion Blur

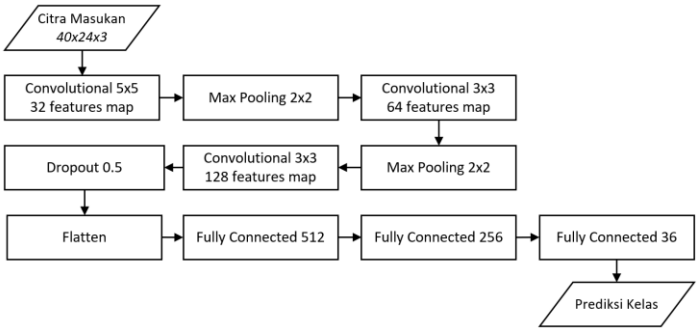
Untuk menyimulasikan kondisi dimana pelat terkadang diambil dalam keadaan kendaraan bergerak, dilakukan penambahan *motion blur* pada karakter. *Motion blur* ditambah dengan mengaplikasikan *filter motion blur* secara horizontal atau

vertikal dengan probabilitas masing-masing 50%. Contoh hasil karakter setelah ditambah *motion blur* dapat dilihat Gambar 3.27.

Gambar 3.27 Hasil karakter dengan *motion blur*

3.2.3 Tahap Pembangunan Arsitektur CNN

Pembangunan model bertujuan untuk menyiapkan *layer*, fungsi aktivasi, *loss function*, dan parameter-parameter yang dibutuhkan. *Convolutional Neural Network* (CNN) yang digunakan terdiri dari *convolutional layer*, *max pooling layer*, *fully connected layer*, dan *dropout layer*. Pembangunan arsitektur CNN mengikuti arsitektur yang dibangun oleh [3] dengan detail yang dapat dilihat pada Gambar 3.28 dan Tabel 3.7.



Gambar 3.28 Arsitektur CNN

Berikut detail arsitektur CNN:

1. Input untuk arsitektur CNN adalah citra karakter berukuran 24x40 *pixels* dengan tiga kanal warna (RGB).
2. Citra masukan akan melalui tiga *Convolutional Layer* dan dua *Max Pooling Layer*.

3. Tiga *Convolutional Layer* yang ada memiliki *filter* dan *kernel* dengan ukuran yang berbeda-beda. Sedangkan ukuran *kernel* dan *stride* pada setiap *Max Pooling Layer* adalah sama yaitu 2x2.
4. Setelah melalui *Convolutional Layer* yang ketiga, citra masukan akan melalui *Dropout Layer* dengan nilai probabilitas 0.5.
5. Setelah melalui *Dropout Layer*, citra masukan akan melalui *Flatten* untuk mengubah matriks menjadi vektor fitur.
6. Setelah itu, vektor fitur akan melalui tiga *Fully Connected Layer* dengan fungsi ReLU pada *Fully Connected Layer* pertama dan kedua dan fungsi Softmax pada *Fully Connected Layer* yang ketiga.
7. Citra masukan yang digunakan memiliki tiga kanal warna (RGB).

Tabel 3.7 Arsitektur CNN

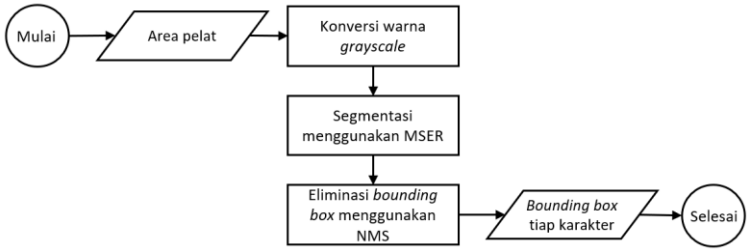
Layer	***Input***	***Output***	**Spesifikasi**
Input Layer	(40, 24, 3)		-
Convolution Layer 1	(40, 24, 3)	(40, 24, 32)	*Filter* : 32
			Kernel : 5x5
			Stride : 1x1
			Fungsi : ReLU
Max Pooling Layer 1	(40, 24, 32)	(20, 12, 32)	*Kernel* : 2x2
			Stride : 2x2
Convolution Layer 2	(20, 12, 32)	(20, 12, 64)	*Filter* : 64
			Kernel : 3x3
			Stride : 1x1
			Fungsi : ReLU
Max Pooling Layer 2	(20, 12, 64)	(10, 6, 64)	*Kernel* : 2x2
			Stride : 2x2
Convolution Layer 3	(10, 6, 64)	(10, 6, 128)	*Filter* : 128
			Kernel : 3x3
			Stride : 1x1
			Fungsi : ReLU
Dropout Layer	(10, 6, 128)	(10, 6, 128)	*Dropout* : 0.5
Flatten	(10, 6, 128)	(7680)	-
Fully Connected Layer 1	(7680)	(512)	Fungsi : ReLU
Fully Connected Layer 2	(512)	(256)	Fungsi : ReLU
Fully Connected Layer 3	(256)	(36)	Fungsi : Softmax

3.2.4 Tahap Deteksi Area Pelat

Pada tahap ini dilakukan deteksi area pelat menggunakan model dari hasil pelatihan *Single Shot Detector* (SSD) yang sudah dijelaskan pada subbab 2.6. Data latih yang digunakan berupa citra pelat sintetis sebanyak 20.000 citra.

3.2.5 Tahap Segmentasi Karakter pada Pelat

Segmentasi karakter dilakukan pada area pelat yang berhasil dideteksi menggunakan metode *Maximally Stable Extremal Regions* (MSER) seperti yang sudah dijelaskan pada subbab 2.7, diagram alir proses segmentasi dapat dilihat pada Gambar 3.29. Sebelum proses segmentasi karakter, dilakukan konversi warna citra pelat ke warna *grayscale* yang kemudian akan dijadikan citra input untuk MSER. Hasil dari segmentasi karakter menggunakan MSER adalah beberapa set *bounding box* untuk tiap karakter. Untuk mengeliminasi *bounding box* yang tumpang tindih sehingga diperoleh satu *bounding box* untuk tiap karakter digunakan metode *Non-Maximum Suppression* (NMS) seperti yang sudah dijelaskan pada subbab 2.8.



Gambar 3.29 Diagram alir segmentasi karakter pada pelat

3.2.6 Tahap Pengenalan Karakter pada Pelat

Akan dilakukan praproses pada citra karakter yang sudah disegmentasi dengan me-*resize* citra menjadi ukuran 24x40 *pixels*. Setelah praproses akan dilakukan klasifikasi kelas citra karakter menggunakan model dari hasil pelatihan *Convolutional Neural*

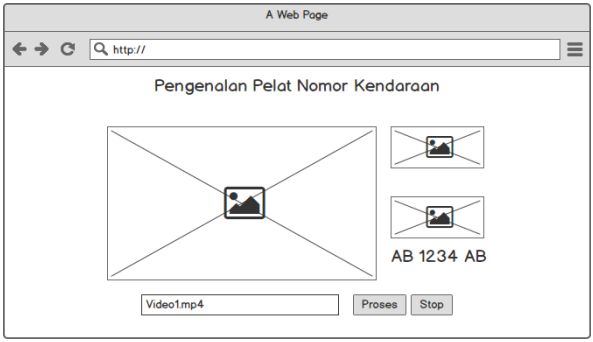
Network (CNN). Data latih yang digunakan berupa citra karakter sintetis sebanyak 54.000 citra.

3.2.7 Tahap Pelatihan dan Evaluasi

Pelatihan *Single Shot Detector* untuk deteksi area pelat dilakukan dengan menggunakan dataset pelat sintetis sebanyak 20.000, *epoch* = 25, dan *batch size* = 8. Sedangkan pada tahap pelatihan CNN, digunakan dataset karakter sintetis sebanyak 54.000 citra yang dibagi sebanyak 70% untuk data latih dan 30% untuk data validasi. Pelatihan CNN dilakukan dengan parameter awal yaitu *epoch* = 200, *batch size* = 512, *learning rate* = 0,0001, dan *optimizer* Adagrad [52]. Akan dilakukan pengujian yang hasilnya akan dievaluasi menggunakan variabel evaluasi akurasi, presisi, *recall*, dan *f1-score*. Data uji yang digunakan merupakan data video yang berasal dari CCTV di lingkungan tempat parkir sepeda motor Teknik Informatika, ITS.

3.3 Desain Umum User Interface

Pada Tugas Akhir ini *Graphical User Interface* (GUI) dibangun menggunakan Node.js sebagai *runtime*, Express.js untuk *routing* URL *browser*, Redis untuk komunikasi antara program Python dengan Express.js, socket.io untuk memperbarui data yang akan ditampilkan pada *browser*, dan HTML untuk *layouting*. *Mockup* GUI dapat dilihat pada Gambar 3.30.



Gambar 3.30 *Mockup* GUI program pengenalan pelat nomor kendaraan

BAB IV
IMPLEMENTASI

Bab ini menjelaskan mengenai implementasi perangkat lunak dari rancangan sistem yang telah dibahas pada Bab 3 yang meliputi kode program. Selain itu, implementasi dari tiap proses, parameter masukan dan keluaran, serta keterangan yang berhubungan dengan program juga akan dijelaskan pada bab ini.

4.1 Lingkungan Implementasi

Dalam mengimplementasikan aplikasi pengenalan pelat nomor kendaraan digunakan beberapa perangkat pendukung sebagai berikut.

4.1.1 Perangkat Keras

Implementasi tugas akhir ini menggunakan desktop *Personal Computer* (PC) MS-7B86. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi AMD Ryzen 7 2700 dengan kecepatan 3.2 GHz, *Random Access Memory* (RAM) sebesar 16 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA GeForce RTX 2080 sebesar 8 GB.

4.1.2 Perangkat Lunak

Perangkat lunak yang digunakan memiliki spesifikasi antara lain menggunakan bahasa pemograman Python 3.6, dilengkapi dengan *library* antara lain OpenCV, Python Imaging Library (PIL), Tensorflow-GPU, Keras-GPU, Numpy, Matplotlib dan Scikit-learn.

4.2 Implementasi Pembuatan Dataset Pelat Sintetis

Pada subbab ini akan dijelaskan mengenai implementasi pembuatan dataset pelat sintetis.

4.2.1 Implementasi Pembuatan Templat Pelat

Langkah pertama dalam pembuatan templat pelat adalah menginisiasi warna hitam dan putih yang diimplementasikan pada Kode Sumber 4.1 pada variabel *black* dan *white* dimana untuk setiap pelat nantinya akan dipilih secara *random* warna hitam (baris 1-4) dan putih (baris 5-9) menggunakan fungsi *numpy.random.randint* dari *library* Numpy.

```
1.  kr = np.random.randint(0, 60)
2.  kg = np.random.randint(0, 60)
3.  kb = np.random.randint(0, 60)
4.  black = (kr, kg, kb)
5.  l = np.random.randint(155, 255)
6.  wr = np.random.randint(0, 30)
7.  wg = np.random.randint(0, 30)
8.  wb = np.random.randint(0, 30)
9.  white = (l+wr, l+wg, l+wb)
```

Kode Sumber 4.1 Fungsi untuk *random* warna hitam dan putih

Langkah selanjutnya adalah membuat dasar pelat dimana pada Tugas Akhir ini dibuat dua jenis dasar pelat, yaitu dengan pinggiran dan tanpa pinggiran. Pembuatan dasar pelat diimplementasikan pada Kode Sumber 4.2 yang memanggil fungsi *round_rectangle* pada Kode Sumber 4.3. Pembuatan persegi panjang memanfaatkan fungsi *Image.new* dari *library* PIL. Pada Kode Sumber 4.3 baris 2 dibuat persegi panjang dengan ukuran dan warna yang sudah ditentukan kemudian pada baris ke 5-6 keempat sudut pada persegi panjang dibuat membulat sehingga mengikuti bentuk pelat yang asli. Baris 11 digunakan untuk memanggil fungsi pembuatan dasar pelat berwarna putih untuk jenis dasar pelat dengan pinggiran, lalu pada baris 12 dibuat dasar pelat berwarna hitam dengan ukuran yang lebih kecil yang akan ditumpuk di atas dasar berwarna putih. Sedangkan untuk jenis pelat tanpa pinggiran langsung dibuat dasar berwarna hitam.

```
1.  plate_base = round_rectangle((plate_base_width, plate_base
    _height), round_corner_radius, white)
2.  plate = round_rectangle((plate_width, plate_height), round
    _corner_radius, black)
3.  plate_base.paste(plate,((plate_base_width - plate_width)//
    2, (plate_base_height - plate_height)//2), plate)
```

Kode Sumber 4.2 Fungsi pembuatan dasar pelat

```
1.  def round_rectangle(size, radius, fill):
2.      width, height = size
3.      rectangle = Image.new('RGBA', size, fill)
4.      corner = round_corner(radius, fill)
5.      rectangle.paste(corner, (0, 0))
6.      rectangle.paste(corner.rotate(90), (0, height - radius
    ))
7.      rectangle.paste(corner.rotate(270), (width - radius, 0
    ))
8.      rectangle.paste(corner.rotate(180), (width - radius, h
    eight - radius))
9.      return rectangle
```

Kode Sumber 4.3 Fungsi pembuatan persegi panjang

Selanjutnya akan ditambahkan tulisan pada dasar pelat yang terdiri dari enam bagian yaitu kode wilayah, nomor polisi, kode akhir wilayah, bulan masa berlaku, dot, dan tahun masa berlaku yang diimplementasikan pada Kode Sumber 4.4. Baris 1-5 merupakan implementasi fungsi untuk *random string* untuk setiap bagian tulisan pada pelat. Baris 7-11 merupakan implementasi fungsi untuk menggambar *string* yang sudah di-*random* pada pelat. Baris 13 dan 14 merupakan pemanggilan fungsi *random_text* dan *draw_text* untuk kode wilayah, baris 16 dan 17 untuk nomor polisi, baris 19 dan 20 untuk kode akhir wilayah, baris 22 dan 23 untuk bulan masa berlaku, baris 25 dan 26 untuk titik (dot), dan terakhir, baris 28 dan 29 untuk tahun masa berlaku.

```
1.  def random_text(string_length):
2.      word = string.ascii_letters
3.      random_word = ''.join(random.choice(word) for i in ran
    ge(string_length))
4.      word = str(random_word).upper()
5.      return word
6.
7.   def draw_text(teks, width, height, font_type):
8.      base = Image.new('RGB', (width, height), color = black
    )
9.      draw = ImageDraw.Draw(base)
10.     draw.text((0, 0), teks, font = font_type, fill = white
    )
11.     return base
12.
13. random_area = random_text(np.random.randint(1, 3))
14. add_plate_area = draw_text(random_area, width, height, fon
    t_type)
15.
16. random_number = str(np.random.randint(1000, 10000))
17. add_plate_number = draw_text(random_number, width, height,
     font_type)
18.
19. random_back = random_text(np.random.randint(1, 3))
20. add_plate_back = draw_text(random_back, width, height, fon
    t_type)
21.
22. random_month = str(np.random.randint(1, 13))
23. add_plate_month = draw_text(random_month, width, height, f
    ont_type)
24.
25. dot = '.'
26. add_plate_dot = draw_text(dot, width, height, font_type)
27.
28. random_year = str(np.random.randint(10, 100))
29. add_plate_year = draw_text(random_year, width, height, fon
    t_type)
```

Kode Sumber 4.4 Fungsi penambahan tulisan pada pelat

4.2.2 Implementasi Penambahan Oklusi pada Pelat

Penambahan oklusi pada pelat diimplementasikan pada Kode Sumber 4.5 dimana akan dibuat persegi panjang yang akan diletakkan pada pelat dengan ukuran serta warna yang dipilih secara *random*. Persegi panjang dibuat dengan memanggil fungsi *round_rectangle* pada Kode Sumber 4.3.

```
1. occlusion_base = round_rectangle((occlusion_base_width, oc
   clusion_base_height), round_corner_radius, occlusion_color
   , position)
2. occlusion = round_rectangle((occlusion_base_width, occlusi
   on_base_height), round_corner_radius, occlusion_color, pos
   ition)
3. occlusion_base.paste(occlusion,(0, 0), occlusion)
```

Kode Sumber 4.5 Fungsi penambahan oklusi pada pelat

4.2.3 Implementasi Penambahan Bayangan pada Pelat

Penambahan bayangan pada pelat diimplementasikan pada Kode Sumber 4.6 dimana warna pelat akan digelapkan dengan memperkalikan tiap nilai kanal warna pelat (RGB) dengan intensitas bayangan yang berkisar antara 10-70%. Intensitas dan ukuran (panjang dan lebar) bayangan akan dipilih secara *random* menggunakan fungsi *random.uniform* untuk tiap pelat

```
1. shadow_intensity = random.uniform(0.1, 0.7)
2. r = int(pixels[j, k][0]*shadow_intensity)
3. g = int(pixels[j, k][1]*shadow_intensity)
4. b = int(pixels[j, k][2]*shadow_intensity)
5. plate_base[j, k] = (r, g, b, plate_base[j, k][3])
```

Kode Sumber 4.6 Fungsi penambahan bayangan pada pelat

4.2.4 Implementasi Penambahan Transparansi pada Pelat

Penambahan penambahan transparansi pada pelat diimplementasikan pada Kode Sumber 4.7 dimana akan dibuat persegi panjang dengan nilai *alpha channel* = 0 yang akan

diletakkan pada pelat dengan ukuran yang dipilih secara *random*. Persegi panjang dibuat dengan memanggil fungsi *round_rectangle* pada Kode Sumber 4.3.

```
1.  transparency_color = (r, g, b, 0)
2.  transparency_base = round_rectangle((transparency_base_wid
    th, transparency_base_height), round_corner_radius, transp
    arency_color, position)
3.  transparency = round_rectangle((transparency_base_width, t
    ransparency_base_height), round_corner_radius, transparenc
    y_color, position)
4.  transparency_base.paste(transparency,(0, 0), transparency)
```

Kode Sumber 4.7 Fungsi penambahan transparansi pada pelat

4.2.5 Implementasi Rotasi Pelat

Rotasi pelat diimplementasikan pada Kode Sumber 4.8 yang memanggil fungsi pada Kode Sumber 4.9 dengan parameter citra pelat, sudut x, sudut y, dan sudut z. Rotasi pelat menggunakan fungsi *transform* dari *library* PIL dengan parameter ukuran citra, *Image.PERSPECTIV*E untuk mengaplikasikan transformasi perspektif, matriks transformasi yang diperoleh dari kode pada baris 2, dan *Image.BICUBIC* yang digunakan untuk interpolasi citra.

```
1.  rotated_img = rotate_along_axis(plate_base, theta, phi, ga
    mma)
```

Kode Sumber 4.8 Fungsi pemanggilan fungsi rotasi untuk pelat

```
1.  def rotate_along_axis(img, theta=0, phi=0, gamma=0, dx=0,
    dy=0, dz=0):
2.      coeffs = make_zero(img, theta, phi, gamma, dx, dy, dz)

3.      return img.transform(size, Image.PERSPECTIVE, coeffs,
    Image.BICUBIC)
```

Kode Sumber 4.9 Fungsi rotasi pelat

4.2.6 Implementasi Penambahan Noise pada Pelat

Penambahan *noise* pada pelat diimplementasikan pada Kode Sumber 4.10 yang memanggil fungsi *add_gaussian_noise* pada Kode Sumber 4.11 dengan parameter citra pelat dan batas *random* standar deviasi. Pada Kode Sumber 4.11 variabel *sigma* pada baris 4 digunakan untuk *random* standar deviasi untuk setiap citra, baris 5 digunakan untuk *generate* distribusi normal dengan parameter input *mean*, standar deviasi, dan ukuran citra, baris 7-8 digunakan untuk merubah objek gambar menjadi *array*, baris 9-11 digunakan untuk menambah *noise* pada tiap *array* karnal warna citra, dan baris 13 digunakan untuk mengembalikan *array* ke objek gambar.

```
1.  plate_base = add_gaussian_noise(plate_base, 10)
```

Kode Sumber 4.10 Fungsi pemanggilan fungsi add_gaussian_noise untuk pelat

```
1.  def add_gaussian_noise(im, var):
2.      mean = 0
3.      sigma = np.random.uniform(0, var)
4.      width, height = im.size
5.      gaussian = np.random.normal(mean, sigma, (height, widt
    h))
6.
7.      img = np.array(im)
8.      img = img.astype('float')
9.      img[..., 0] = img[..., 0] + gaussian
10.     img[..., 1] = img[..., 1] + gaussian
11.     img[..., 2] = img[..., 2] + gaussian
12.
13.     new_img = Image.fromarray(normalize(img).astype('uint8
    '),'RGBA')
14.     return new_img
```

Kode Sumber 4.11 Fungsi penambahan *noise*

4.2.7 Implementasi Penambahan Motion Blur pada Pelat

Penambahan *motion blur* diimplementasikan pada Kode Sumber 4.12 yang memanggil fungsi *add_motion_blur* pada Kode Sumber 4.13 dengan parameter citra pelat dan ukuran *kernel*. *Blur* dapat diaplikasikan secara horizontal atau vertikal dengan probabilitas masing-masing 50%. Pada Kode Sumber 4.13 baris 5 dan 7 digunakan untuk membuat *kernel motion blur* dengan ukuran 3x3 dimana baris 5 digunakan untuk *blur* secara horizontal dan baris 7 secara vertikal. Baris 12 digunakan untuk mengaplikasikan *kernel motion blur* pada citra menggunakan fungsi *filter* dari *library* PIL.

```
1.  plate_base = add_motion_blur(img, 3)
```

Kode Sumber 4.12 Fungsi pemanggilan fungsi add_motion_blur untuk pelat

```
1.  def add_motion_blur(img, size):
2.      kernel_motion_blur = np.zeros((size, size))
3.
4.      if random.random() <= 0.5: #horizontal
5.          kernel_motion_blur[int((size-
    1)/2), :] = np.ones(size)
6.      else:  #vertikal
7.          kernel_motion_blur[:, int((size-
    1)/2)] = np.ones(size)
8.
9.      kernel_motion_blur = kernel_motion_blur / size
10.     kernel_motion_blur = tuple(kernel_motion_blur.reshape(
    1, -1)[0])
11.
12.     img = img.filter(ImageFilter.Kernel((size, size), kern
    el_motion_blur, 1, 0))
13.
14.     return img
```

Kode Sumber 4.13 Penambahan *motion blur*

4.2.8 Implementasi Penyatuan Pelat dengan Background

Penyatuan pelat dengan *background* diimplementasikan pada Kode Sumber 4.14 menggunakan fungsi *paste* dari *library* PIL dimana parameternya adalah citra yang ingin ditempel pada *background* dan posisi dimana citra pelat akan diletakkan di bagian tengah *background*. Baris 1-4 digunakan untuk menghitung posisi dimana citra pelat akan ditempel pada *background*.

```
1.  new_width, new_height = plate_base.size
2.  back_width, back_height = back.size
3.  x = (back_width - new_width)//2
4.  y = (back_height - new_height)//2
5.  back.paste(plate_base, (x, y), plate_base)
```

Kode Sumber 4.14 Fungsi penyatuan pelat dengan *background*

4.2.9 Implementasi Penambahan Motion Blur

Penambahan *motion blur* diimplementasikan pada Kode Sumber 4.15 yang memanggil fungsi *add_motion_blur* pada Kode Sumber 4.13 dengan parameter citra pelat dan ukuran kernel.

```
1.  back = add_motion_blur(back, 3)
```

Kode Sumber 4.15 Fungsi pemanggilan fungsi add_motion_blur untuk citra pelat dan *background*

4.2.10 Implementasi Penambahan Blur

Penambahan blur pada citra diimplementasikan pada Kode Sumber 4.16 menggunakan fungsi *ImageFilter.GaussianBlur* dari *library* PIL dengan parameter radius *blur*. Untuk setiap citra nilai radius *blur* = 1.

```
1.  back = back.filter(ImageFilter.GaussianBlur(radius = 1))
```

Kode Sumber 4.16 Fungsi penambahan blur

4.2.11 Implementasi Penambahan Noise

Penambahan *noise* pada pelat dan *background* yang sudah disatukan diimplementasikan pada Kode Sumber 4.17 yang memanggil fungsi *add_gaussian_noise* pada Kode Sumber 4.11 dengan parameter citra pelat dan *background* serta batas *random* standar deviasi.

```
1.  back = add_gaussian_noise(back, 8)
```

Kode Sumber 4.17 Fungsi pemanggilan fungsi add_gaussian_noise untuk citra pelat dan *background*

4.2.12 Implementasi Crop Citra

Pemotongan atau *crop* citra diimplementasikan pada Kode Sumber 4.18 dimana baris 1-9 digunakan untuk menghitung batasan posisi awal *crop* sehingga seluruh bagian pelat akan ada di dalam citra setelah di-*crop*. Baris 11 digunakan untuk *crop* citra dengan parameter koordinat awal (x, y) dan koordinat akhir (x+w, y+h).

```
1.  w, h = 533, 300
2.  min_xl = row_cells[1].value
3.  min_yl = row_cells[2].value
4.  max_xl = row_cells[3].value
5.  max_yl = row_cells[4].value
6.  x_min = max_xl - 384
7.  x_max = min_xl
8.  y_min = max_yl - 246
9.  y_max = min_yl
10.
11. gambar = positive_img[y:y+h, x:x+w]
```

Kode Sumber 4.18 Fungsi *crop* citra

4.3 Implementasi Pembuatan Dataset Karakter Sintetis

Pada subbab ini akan dijelaskan mengenai implementasi pembuatan dataset karakter sintetis.

4.3.1 Implementasi Pembuatan Karakter

Pembuatan karakter diimplementasikan pada Kode Sumber 4.19 yang memanggil fungsi *add_character* pada Kode Sumber 4.20 dimana fungsi *ImageDraw.Draw* dari *library* PIL digunakan untuk menambahkan teks karakter pada *background* yang diinisiasi pada baris 3.

```
1.  letter = add_character(str(i))
```

Kode Sumber 4.19 Fungsi pemanggilan fungsi penambahan karakter

```
1.  def add_character(black, white, word):
2.      w, h = font_big.getsize(word)
3.      letter = Image.new('RGBA', (w, h), color = black)
4.      draw = ImageDraw.Draw(letter)
5.      draw.text((0, 0), word, font = font_big, fill = white)

6.      return letter
```

Kode Sumber 4.20 Fungsi pembuatan karakter

4.3.2 Implementasi Rotasi Karakter

Rotasi pelat diimplementasikan pada Kode Sumber 4.21 yang memanggil fungsi *rotate_along_axis* pada Kode Sumber 4.9 dengan parameter citra pelat, besar sudut x, sudut y, dan sudut z.

```
1.  rotated_img = rotate_along_axis(plate_base, theta, phi, ga
    mma)
```

Kode Sumber 4.21 Fungsi pemanggilan fungsi rotasi untuk karakter

4.3.3 Implementasi Pembuatan Background

Pembuatan *background* diimplementasikan pada Kode Sumber 4.22 dengan menggunakan fungsi *Image.new* dari *library* PIL yang memiliki tiga parameter yaitu mode warna, ukuran, dan warna gambar.

```
1.  back = Image.new('RGB', (24, 40), color = black)
```

Kode Sumber 4.22 Fungsi pembuatan *background* untuk karakter

4.3.4 Implementasi Penyatuan Karakter dengan Background

Penyatuan karakter dengan *background* diimplementasikan pada Kode Sumber 4.23 menggunakan fungsi *paste* dari *library* PIL pada baris 5 dimana parameternya adalah citra yang ingin ditempel pada *background*, posisi dimana citra pelat akan diletakkan di bagian tengah *background*, dan *mask* yang sama dengan citra karakter. Baris 1-4 digunakan untuk menghitung posisi dimana citra karakter akan ditempel pada *background*.

```
1.  new_width, new_height = letter.size
2.  back_width, back_height = back.size
3.  x = (back_width - new_width)//2
4.  y = (back_height - new_height)//2
5.  back.paste(letter, (x, y), letter)
```

Kode Sumber 4.23 Fungsi penyatuan karakter dengan *background*

4.3.5 Implementasi Penambahan Motion Blur

Penambahan *motion blur* diimplementasikan pada Kode Sumber 4.24 yang memanggil fungsi add_motion_blur pada Kode Sumber 4.13 dengan parameter citra karakter dan ukuran kernel.

```
1.  plate_base= add_motion_blur(plate_base, 3)
```

Kode Sumber 4.24 Fungsi pemanggilan fungsi add_motion_blur untuk citra karakter dan *background*

4.4 Implementasi Pembangunan Arsitektur SSD

Pada subbab ini akan dijelaskan fungsi-fungsi untuk membangun arsitektur SSD yang sudah dijelaskan pada subbab 2.6 dimulai dengan *load* arsitektur VGG16 yang diimplementasikan pada Kode Sumber 4.25 dimana pada baris 3-11 digunakan untuk

load pre-trained model VGG16 yang sudah diunduh sebelumnya. Selanjutnya set *layer* tambahan yang sudah dijelaskan pada subbab 2.6 dimplementasikan pada Kode Sumber 4.26.

```
1.  def __load_vgg(self, vgg_dir):
2.      sess = self.session
3.      graph = tf.saved_model.loader.load(sess, ['vgg16'], vg
    g_dir+'/vgg')
4.      self.image_input = sess.graph.get_tensor_by_name('imag
    e_input:0')
5.      self.keep_prob   = sess.graph.get_tensor_by_name('keep
    _prob:0')
6.      self.vgg_conv4_3 = sess.graph.get_tensor_by_name('conv
    4_3/Relu:0')
7.      self.vgg_conv5_3 = sess.graph.get_tensor_by_name('conv
    5_3/Relu:0')
8.      self.vgg_fc6_w   = sess.graph.get_tensor_by_name('fc6/
    weights:0')
9.      self.vgg_fc6_b   = sess.graph.get_tensor_by_name('fc6/
    biases:0')
10.     self.vgg_fc7_w   = sess.graph.get_tensor_by_name('fc7/
    weights:0')
11.     self.vgg_fc7_b   = sess.graph.get_tensor_by_name('fc7/
    biases:0')
```

Kode Sumber 4.25 Fungsi *load*

```
1.  def __build_ssd_layers(self):
2.           stride10 = 1
3.          padding10 = 'VALID'
4.          if len(self.preset.maps) >= 7:
5.              stride10 = 2
6.              padding10 = 'SAME'
7.
8.          x, l2  = conv_map(self.mod_conv7,    256, 1, 1, 'c
    onv8_1')
9.          self.ssd_conv8_1 = self.__with_loss(x, l2)
10.         x, l2 = conv_map(self.ssd_conv8_1,  512, 3, 2, 'co
    nv8_2')
11.         self.ssd_conv8_2 = self.__with_loss(x, l2)
```

```
12.         x, l2  = conv_map(self.ssd_conv8_2,  128, 1, 1, 'c
    onv9_1')
13.         self.ssd_conv9_1 = self.__with_loss(x, l2)
14.         x, l2 = conv_map(self.ssd_conv9_1,  256, 3, 2, 'co
    nv9_2')
15.         self.ssd_conv9_2 = self.__with_loss(x, l2)
16.         x, l2 = conv_map(self.ssd_conv9_2,  128, 1, 1, 'co
    nv10_1')
17.         self.ssd_conv10_1 = self.__with_loss(x, l2)
18.         x, l2 = conv_map(self.ssd_conv10_1, 256, 3, stride
    10, 'conv10_2', padding10)
19.         self.ssd_conv10_2 = self.__with_loss(x, l2)
20.         x, l2 = conv_map(self.ssd_conv10_2, 128, 1, 1, 'co
    nv11_1')
21.         self.ssd_conv11_1 = self.__with_loss(x, l2)
22.         x, l2 = conv_map(self.ssd_conv11_1, 256, 3, 1, 'co
    nv11_2', 'VALID')
23.         self.ssd_conv11_2 = self.__with_loss(x, l2)
24.
25.         if len(self.preset.maps) < 7:
26.             return
27.
28.         x, l2 = conv_map(self.ssd_conv11_2, 128, 1, 1, 'co
    nv12_1')
29.         paddings = [[0, 0], [0, 1], [0, 1], [0, 0]]
30.         x = tf.pad(x, paddings, "CONSTANT")
31.         self.ssd_conv12_1 = self.__with_loss(x, l2)
32.         x, l2 = conv_map(self.ssd_conv12_1, 256, 3, 1, 'co
    nv12_2', 'VALID')
33.         self.ssd_conv12_2 = self.__with_loss(x, l2)
```

Kode Sumber 4.26 Fungsi pembangunan arsitektur SSD

4.5 Implementasi Pembangunan Arsitektur CNN

Pada subbab ini akan dijelaskan fungsi-fungsi untuk membangun arsitektur CNN yang diimplementasikan pada Kode Sumber 4.27 dimana baris 1-2 digunakan untuk inisialisasi ukuran citra input. Pada baris 4 dilakukan pemanggilan fungsi *build* pada Kode Sumber 4.28 untuk membangun arsitektur CNN.

```
1.  img_width, img_height = 24, 40
2.  no_of_channels = 3
3.  #inisialisasi model
4.  model = Net.build(height = img_height, width = img_width,
    depth = no_of_channels)
```

Kode Sumber 4.27 Fungsi pembangunan arsitektur CNN

```
1.  def build(width, height, depth):
2.          model = Sequential()
3.
4.          model.add(Convolution2D(32, (5, 5), input_shape =
    (height, width, depth), padding='same'))
5.          model.add(Activation('relu'))
6.          model.add(MaxPooling2D(pool_size = (2, 2)))
7.
8.          model.add(Convolution2D(64, (3, 3),padding='same')
    )
9.          model.add(Activation('relu'))
10.         model.add(MaxPooling2D(pool_size = (2, 2)))
11.
12.         model.add(Convolution2D(128, (3, 3), padding='same
    '))
13.         model.add(Activation('relu'))
14.
15.         model.add(Dropout(0.5))
16.
17.         model.add(Flatten())
18.
19.         model.add(Dense(512))
20.         model.add(Activation('relu'))
21.
22.         model.add(Dense(256))
23.         model.add(Activation('relu'))
24.
25.         model.add(Dense(36))
26.         model.add(Activation('softmax'))
27.
28.         return model
```

Kode Sumber 4.28 Fungsi penyusunan jaringan arsitektur CNN

Arsitektur yang dibangun terdiri dari lima macam *layer* antara lain *Convolution2D, MaxPooling2D, Flatten, Dropout*, dan *Dense*. *Convolution2D* adalah implementasi *Convolution Layer* dalam bentuk dua dimensi. Terdapat lima parameter yang digunakan pada *Convolution2D* dengan penjelasan sebagai berikut:

1. Parameter *filters* adalah jumlah *filter*.
2. Parameter *kernel_size* adalah ukuran *kernel*.
3. Parameter *strides* adalah ukuran *stride* yang digunakan.
4. Parameter *acitivation* adalah fungsi aktivasi, terdapat beberapa pilihan fungsi aktivasi yang disediakan oleh Keras, salah satunya adalah 'relu' untuk mengimplementasikan fungsi aktivasi Rectified Linear Unit (ReLU).
5. Parameter *padding* menentukan apakah *output* diaplikasikan *Zero Padding*. Jika *padding* bernilai '*valid*' maka *output* tidak diaplikaskan *Zero Padding* dan jika bernilai '*same'* maka bentuk keluaran akan diaplikasikan *Zero Padding* untuk menjaga keluaran sama dengan *input*. Nilai *default padding* adalah *'valid'.*

MaxPooling2D adalah implementasi *Max Pooling Layer* dalam bentuk dua dimensi. Terdapat dua parameter yang digunakan pada *MaxPooling2D* dengan penjelasan sebagai berikut:

1. Parameter *pool_size* adalah ukuran *kernel Pooling Layer*.
2. Parameter *strides* adalah ukuran *stride* yang digunakan. Nilai default *strides* jika tidak dicantumkan adalah *'None'* yang berarti ukuran *stride* akan disamakan dengan *pool_size*.

Flatten pada arsitektur CNN biasa digunakan ketika akan memasuki *Fully Connected Layer*. Hal ini karena *Flatten* berfungsi untuk merubah keluaran yang sebelumnya berupa *feature map*

menjadi *feature vector*, sehingga dapat dijadikan *input* pada *Fully Connected Layer*.

Dropout adalah teknik regularisasi dimana beberapa *neuron* akan dipilih secara acak dan tidak dipakai selama proses pelatihan. *Dropout* dapat digunakan untuk mencegah terjadinya *overfitting* dan mempercepat proses *learning*. *Dropout* memiliki parameter *rate* yakni nilai probabilitas yang dipakai dalam menentukan *neuron* yang akan dipilih untuk tidak dipakai.

Dense adalah implementasi *Fully Connected Layer*. Terdapat dua parameter yang digunakan pada *Dense* dengan penjelasan sebagai berikut:

1. Parameter *units* adalah jumlah neuron.
2. Parameter *acitivation* adalah fungsi aktivasi, terdapat beberapa pilihan fungsi aktivasi yang disediakan oleh Keras, salah satunya adalah 'relu' untuk mengimplementasikan fungsi aktivasi Rectified Linear Unit (ReLU) dan 'softmax' untuk mengimplementasikan fungsi Softmax yang biasa digunakan pada akhir arsitektur untuk klasifikasi label kelas.

4.6 Implementasi Deteksi Area Pelat

Deteksi area pelat dilakukan dengan menggunakan *Single Shot Detector* (SSD) yang sudah dibangun sebelumnya. Deteksi area pelat diimplementasikan pada Kode Sumber 4.29 dimana baris 1-2 digunakan untuk menyimpan *frame* video ke dalam *array* untuk dijadikan masukan ke arsitektur SSD. Baris 4 digunakan untuk memprediksi lokasi pelat yang ada pada *frame* video.

```
1.  batch.append(img)
2.  batch = np.array(batch)
3.  feed = {img_input: batch}
4.  enc_boxes = sess.run(result, feed_dict=feed)
```

Kode Sumber 4.29 Fungsi SSD

4.7 Implementasi Segmentasi Karakter pada Pelat

Sebelum masuk ke proses segmentasi karakter, dilakukan konversi warna citra area pelat ke warna *grayscale* yang diimplementasikan pada Kode Sumber 4.30. Konversi warna dilakukan dengan menggunakan fungsi *cv2.cvtColor* dari *library* OpenCV dimana parameter yang digunakan adalah area pelat sebagai citra masukan dan *cv2.COLOR_BGR2GRAY* sebagai mode perubahan kanal, dimana BGR merupakan kanal awal citra.

```
1.  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Kode Sumber 4.30 Fungsi konversi warna *grayscale*

Setelah konversi warna, selanjutnya dilakukan segmentasi karakter pada area pelat yang diimplementasikan pada Kode Sumber 4.31. Fungsi yang digunakan adalah *cv2.MSER_create* dari *library* OpenCV dengan satu parameter *_delta* = 3 yang merupakan perbandingan dari $\frac{(size_i - size_{i-delta})}{size_{i-delta}}$.

```
1.  mser = cv2.MSER_create(_delta = 3)
2.  regions, rects = mser.detectRegions(gray)
3.  min_height = 20/100*height
4.  max_height = 45/100*height
5.  min_width = 3/100*width
6.  max_position = 62/100*height
7.  max_position_x = 94/100*width
8.  bounding_boxes = np.array([], dtype=np.int64).reshape(0,4)
9.  for (x, y, w, h) in rects:
10.     if w < h and h > min_height and h < max_height and (y+
    h) < max_position_y and w > min_width and y > 5 and (x) <
    max_position_x:
11.         bounding_boxes = np.vstack([bounding_boxes, np.arr
    ay([[x, y, x+w, y+h]])])
```

Kode Sumber 4.31 Fungsi segmentasi karakter

Fungsi *detectRegions* pada baris 7 memiliki satu parameter yaitu citra masukan. Pada baris 9-11 dilakukan seleksi area terbesar

dan terkecil *bounding box* dengan batasan yang ditentukan pada kode sumber baris 3-7, batasan yang diperoleh dari percobaan yang sudah dilakukan antara lain:

1. Lebar *bounding box* lebih besar dari panjangnya.
2. Lebar *bounding box* tidak kurang dari 20% dan tidak lebih dari 45% lebar citra pelat yang berhasil dideteksi.
3. Panjang *bounding box* tidak kurang dari 3% panjang citra pelat yang berhasil dideteksi.
4. Koordinat x_1 *bounding box* tidak lebih dari 94% lebar citra pelat yang berhasil dideteksi.
5. Koordinat y_1 *bounding box* tidak kurang dari 5 *pixel*.
6. Koordinat y_2 *bounding box* tidak lebih dari 62% lebar citra pelat yang berhasil dideteksi.

Selanjutnya akan dilakukan reduksi *bounding box* yang tumpang tindih menggunakan metode *Non-Maximum Supression* (NMS) yang diimplementasikan pada Kode Sumber 4.32 dengan memanggil fungsi *non_max_supression_fast* yang memiliki dua parameter input yaitu set *bounding boxes* dan *threshold* area yang *overlap* sama dengan 0,3.

```
1.  pick = non_max_suppression_fast(bounding_boxes, 0.3)
```

Kode Sumber 4.32 Fungsi *Non-Maximum Supression*

4.8 Implementasi Pengenalan Karakter pada Pelat

Pengenalan karakter diimplementasikan pada Kode Sumber 4.33 dimana pada baris 1 dilakukan *sorting bounding box* berdasarkan koordinat x terkecil untuk mengurutkan *bounding box* karakter pada pelat dari kiri ke kanan.

Pada baris 2-6, karakter pada pelat di-*crop* sesuai dengan *bounding box* masing-masing dan dilakukan praproses dengan *resize* citra karakter menjadi 24x40 *pixels*. Baris 8-9 digunakan untuk *reshape array* citra karakter sehingga dapat dijadikan input untuk arsitektur CNN. Baris 10 digunakan untuk memprediksi kelas citra karakter menggunakan model dari proses pelatihan

CNN. Baris 11-12 digunakan untuk merubah kelas hasil prediksi yang sebelumnya berupa kelas *integer* (0-35) ke kelas karakter (0-9, A-Z). Baris 13 digunakan untuk menyatukan semua karakter ke dalam bentuk *string*.

```
1.  pick = pick[np.argsort(pick[:, 0])[::1]]
2.  for (x1, y1, x2, y2) in pick:
3.      batas = 1
4.      cropped_region = img[y1-batas:y2, x1-batas:x2]
5.      cropped_region = cv2.resize(cropped_region, (24,40))
6.      crop_images.append(cropped_region)
7.
8.  crop_images= np.reshape(crop_images, (len(crop_images), 40
    , 24, 3))
9.  x_pred = np.array(crop_images)
10. y_pred = model.predict_classes(x_pred)
11. for idx, element in enumerate(y_pred):
12.     characters.append(labels[element])
13. plate_full = convert(characters)
```

Kode Sumber 4.33 Fungsi pengenalan karakter pada pelat

4.9 Implementasi Pelatihan dan Evaluasi

Setelah membangun arsitektur *Convolutional Neural Network*, dilakukan proses pelatihan dan evaluasi. Proses pelatihan dijalankan dengan jumlah *epoch* = 200, ukuran *batch* = 512, *optimizer* Adagrad dengan *learning rate* = 0,0001, dan *loss function* berupa *Cross Entropy* yang diimplementasikan pada Kode Sumber 4.34. Pelatihan dilakukan dengan menggunakan fungsi *fit* dari *library* Keras yang diimplementasikan pada Kode Sumber 4.35 dimana parameter inputnya adalah data latih dan label kelasnya, ukuran *batch*, jumlah *epoch*, dan *validation_data* yang merupakan data validasi dan label kelasnya. Proses pelatihan akan menghasilkan model untuk klasifikasi karakter.

Setelah proses pelatihan, model akan diuji dengan menggunakan data uji yang diimplementasikan pada Kode Sumber 4.36 menggunakan fungsi *evaluate* dari *library* Keras dengan

parameter masukan berupa data uji dan label kelasnya. Dari fungsi *evaluate* akan didapatkan nilai akurasi dan *loss* model terhadap data uji. Untuk mendapatkan nilai presisi, *recall*, dan *f1-score* dilakukan prediksi kelas dari data uji menggunakan fungsi *predict* dari *library* Keras yang diimplementasikan pada Kode Sumber 4.37. Pada baris 1, variabel label digunakan untuk menyimpan label kelas data uji dan pada baris 2, variabel *predictions* digunakan untuk menyimpan hasil prediksi kelas dari data uji menggunakan fungsi *predict* dari *library* Keras. Hasil prediksi kelas masih berupa kelas kategorikal yang akan diubah ke tipe kelas numerik menggunakan fungsi *argmax* dari *library* Numpy pada baris 3. Pada baris 12 digunakan fungsi *classification_report* dari *library* Scikit-learn untuk mengevaluasi hasil prediksi kelas terhadap kelas yang sebenarnya untuk masing-masing kelas sehingga didapatkan nilai presisi, *recall*, dan *f1-score.*

```
1.  epochs = 200
2.  batch_size = 512
3.  #compile model
4.  optimizer = optimizers.Adagrad(lr=0.0001)
5.  model.compile(loss='categorical_crossentropy',
6.                optimizer=optimizer,
7.                metrics=['accuracy'])
```

Kode Sumber 4.34 Fungsi *compile* arsitektur CNN

```
1.  history = model.fit(
2.            x_train, y_train,
3.            batch_size=batch_size,
4.            epochs=epochs,
5.            validation_data=(x_val, y_val))
```

Kode Sumber 4.35 Fungsi pelatihan arsitektur CNN

```
evaluate = model.evaluate(x_test, y_test)
print('Accuracy', evaluate[1])
print('Loss', evaluate[0])
```

Kode Sumber 4.36 Fungsi evaluasi akurasi dan loss arsitektur CNN

```
label = test['label'].values
predictions = model.predict(x_test)
predicted_classes = np.argmax(predictions, axis=1)
report = classification_report(label, predicted_classes)
```

Kode Sumber 4.37 Fungsi evaluasi presisi, *recall*, dan *f1-score*

BAB V
UJI COBA DAN EVALUASI

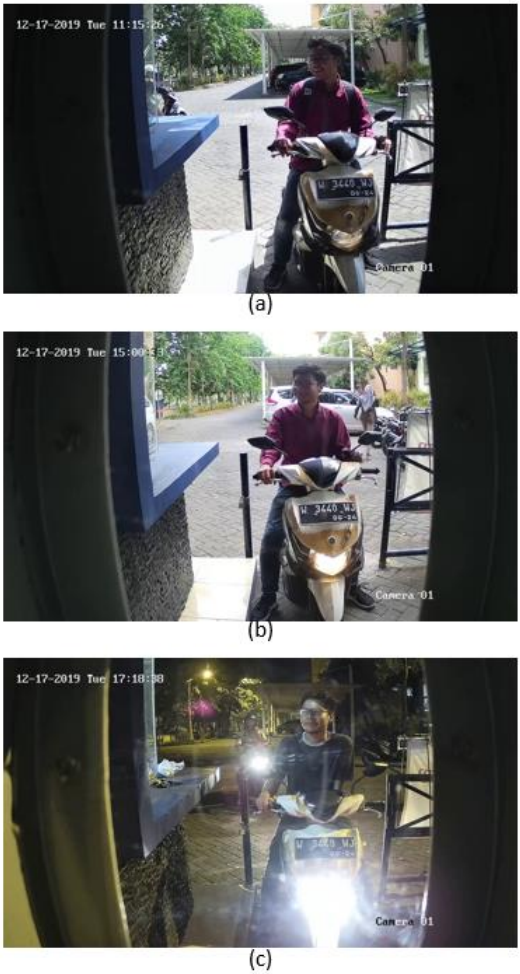
Bab ini akan membahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba pada tugas akhir ini adalah sebuah desktop *personal computer* (PC) MS-78B6. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi perangkat keras AMD Ryzen 7 2700 dengan kecepatan 3,2 GHz, *Random Access Memory* (RAM) sebesar 16 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA GeForce RTX 2080 sebesar 8 GB. Pada sisi perangkat lunak, uji coba pada tugas akhir ini dilakukan dengan menggunakan bahasa pemograman Python 3.6 dilengkapi dengan *library* antara lain Keras, Tensorflow, Python Imaging Library (PIL), OpenCV, Numpy, Matplotlib dan Scikit-learn.

5.2 Dataset

Pada Tugas Akhir ini, dataset yang digunakan pada saat proses pelatihan arsitektur SSD berupa citra pelat sintetis dan pada saat proses pelatihan arsitektur CNN menggunakan citra karakter sintetis. Sedangkan dataset yang digunakan pada saat proses pengujian berupa data video yang berasal dari CCTV di lingkungan tempat parkir sepeda motor Teknik Informatika, ITS. Spesifikasi dataset pengujian dapat dilihat pada Tabel 5.1. Beberapa contoh data uji dapat dilihat pada Gambar 5.1.



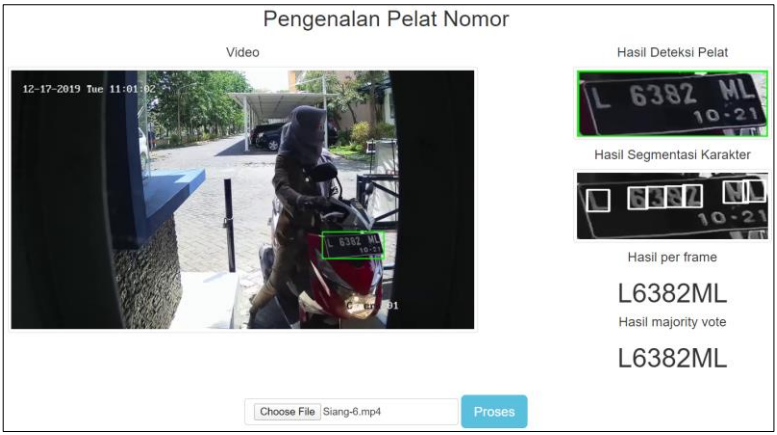
Gambar 5.1 Contoh data uji berupa video. (a) Waktu pengambilan siang hari; (b) Waktu pengambilan sore hari; (c) Waktu pengambilan malam hari.

Tabel 5.1 Spesifikasi dataset pengujian

Keterangan	Spesifikasi
Ukuran frame	1920x1080 *pixels*
Jumlah video	30
Jumlah waktu pengambilan	3 (siang, sore, malam)
Jumlah video per-waktu pengambilan	10
Ukuran file	1.575-4.031 KB
Ekstensi	.mp4
Jumlah *frame* per video	20 *frames* (kecuali video 2 waktu pengambilan sore sebanyak 15 *frames* dan video 8 waktu pengambilan malam sebanyak 13 *frames*) didapatkan dengan membaca *frame* dengan kelipatan 7.
Kanal warna	3 (RGB)

5.3 Hasil Pengujian Seluruh Proses

Hasil pengujian seluruh proses dapat dilihat pada Gambar 5.2. Pengujian dimulai dari deteksi area pelat pada tiap *frame* video menggunakan *Single Shot Detector* (SSD). Setiap pelat yang berhasil dideteksi akan melalui tahap praproses dengan konversi warna ke *grayscale* sebelum masuk ke proses segmentasi karakter. Segmentasi karakter dilakukan menggunakan metode *Maximally Stable Extremal Regions* (MSER). Tiap hasil segmentasi karakter akan diklasifikasi kelasnya menggunakan *Convolutional Neural Network* (CNN). Hasil dari keseluruhan proses berupa *string* karakter pada pelat nomor per *frame* dan hasil *majority vote*.



Gambar 5.2 Hasil pengujian seluruh proses

5.4 Skenario Uji Coba

Proses uji coba berguna untuk identifikasi parameter-parameter yang menghasilkan performa model deteksi area pelat menggunakan SSD dan pengenalan karakter menggunakan CNN yang paling optimal. Parameter yang tepat akan memberikan hasil yang baik pada saat uji coba. Hasil terbaik dari suatu skenario uji coba akan digunakan untuk skenario uji coba berikutnya. Terdapat empat skenario yang akan diujicobakan yaitu:

1. Uji coba kondisi pada citra pelat sintetis
2. Uji coba parameter *optimizer* pada arsitektur CNN
3. Uji coba pada data gambar
4. Uji coba pada data video

Tabel 5.2 dan Tabel 5.3 berisi parameter awal untuk arsitektur SSD dan CNN yang dapat berubah pada setiap uji coba yang dilakukan. Pada setiap skenario uji coba akan ditetapkan nilai parameter yang dapat meningkatkan kinerja arsitektur.

Tabel 5.2 Parameter awal yang digunakan dalam arsitektur SSD

Keterangan	Parameter
Jumlah *epoch*	25
Ukuran *batch*	8
SSD network	VGG300
Kondisi citra pelat sintetis	Semua kondisi

Tabel 5.3 Parameter awal yang digunakan dalam arsitektur CNN

Keterangan	Parameter
Jumlah *epoch*	200
Ukuran *batch*	512
Optimizer	Adagrad
Learning Rate	0,0001
Loss function	*Categorical Cross Entropy*

5.4.1 Uji Coba Kondisi pada Citra Pelat Sintetis

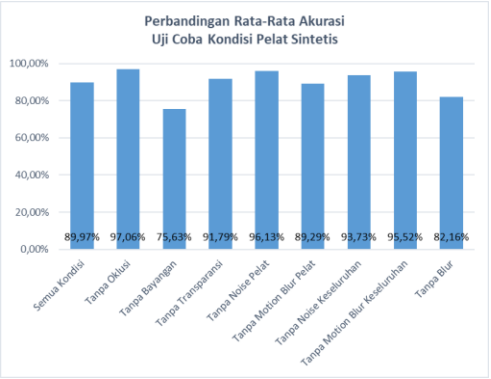
Uji coba kondisi citra pelat sintetis terhadap data uji berupa video digunakan untuk mengetahui kondisi mana saja yang menghasilkan performa terbaik dari model deteksi area pelat menggunakan SSD. Kondisi yang akan diujicobakan antara lain oklusi, bayangan, transparansi, *noise* pada pelat, *motion blur* pada pelat, *noise* keseluruhan, *motion blur* keseluruhan, dan *blur* yang masing-masing akan dinonaktifkan secara bergantian pada proses uji coba. Semua kondisi meliputi oklusi, bayangan, transparansi, *noise* pada pelat, *motion blur* pada pelat, *noise* keseluruhan, *motion blur* keseluruhan, dan *blur* diaktifkan juga akan diujicobakan.

Akurasi lokalisasi area pelat didapatkan dengan menghitung secara manual *true positive* dan *true negative* pada tiap *frame* data video. Hasil lokalisasi area pelat dinilai benar 100% apabila *bounding box* hasil lokalisasi mencakup seluruh karakter pada area pelat. Uji coba kondisi pada citra pelat sintetis menghasilkan perbandingan rata-rata akurasi pada proses pengujian yang dapat dilihat pada Tabel 5.4 dan Gambar 5.3. Didapatkan hasil evaluasi rata-rata akurasi terbaik sebesar 97,06% dimana kondisi oklusi dinonaktifkan. Didapatkan juga rata-rata akurasi terbaik sebesar

99% saat pengambilan video pada sore hari. Dari hasil uji coba maka model deteksi area pelat yang dilatih menggunakan dataset pelat sintetis dengan kondisi oklusi dinonaktifkan digunakan dalam uji coba selanjutnya.

Tabel 5.4 Perbandingan akurasi pada uji coba kondisi citra pelat sintetis

Kondisi	**Akurasi (%)**			
	Siang	**Sore**	**Malam**	**Rata-Rata**
Semua kondisi	95,00	94,50	80,42	89,97
Tanpa oklusi	98,00	**99,00**	94,19	**97,06**
Tanpa bayangan	85,00	73,50	68,38	75,63
Tanpa transparansi	97,50	93,50	84,38	91,79
Tanpa *noise* pada pelat	**98,50**	98,50	91,38	96,13
Tanpa *motion blur* pada pelat	93,00	91,17	83,69	89,29
Tanpa *noise* keseluruhan	98,00	92,00	91,19	93,73
Tanpa *motion blur* keseluruhan	97,50	94,33	**94,73**	95,52
Tanpa *blur*	96,50	79,33	70,65	82,16



Gambar 5.3 Grafik perbandingan akurasi pada uji coba kondisi citra pelat sintetis

5.4.2 Uji Coba Parameter Optimizer pada Arsitektur CNN

Uji coba parameter *optimizer* pada arsitektur CNN digunakan untuk mengetahui *optimizer* mana yang menghasilkan performa terbaik dari model klasifikasi karakter menggunakan CNN. *Optimizer* yang akan diujicobakan antara lain Adagrad, SGD, RMSprop, dan Adam. Percobaan dilakukan dengan *learning rate* 0,0001 dan menggunakan data validasi sebanyak 16.200 atau 30% dari total dataset sedangkan sebanyak 37.800 atau 70% digunakan sebagai data latih.

Uji coba *optimizer* pada arsitektur CNN menghasilkan perbandingan lama waktu pelatihan, nilai akurasi, *weighted average precision*, *weighted average recall*, dan *weighted average f1-score* yang dapat dilihat pada Tabel 5.5. Hasilnya, didapatkan *optimizer* Adam merupakan *optimizer* yang paling optimal dengan akurasi sebesar 99,88%.

Tabel 5.5 Perbandingan hasil evaluasi pada uji coba parameter *optimizer* pada arsitektur CNN

Optimizer	Adagrad	SGD	Adam	RMSprop
Lama Waktu Pelatihan (detik)	376,21	**354,89**	392,77	383,96
Akurasi (%)	99,85	9,99	**99,88**	99,84
Loss	0,0078	3,5289	**0,0071**	0,0156
Weighted Average Precision (%)	99,85	6,85	**99,89**	99,85
Weighted Average Recall (%)	99,85	9,99	**99,89**	99,85
Weighted Average F1-Score (%)	99,85	4,88	**99,89**	99,85

5.4.3 Uji Coba pada Data Video

Uji coba dilakukan untuk mengukur performa sistem yang telah dibuat pada data video kendaraan sepeda motor yang akan dikenali karakter pada pelat nomornya. Evaluasi performa segmentasi dan pengenalan karakter dilakukan dengan membandingkan secara visual hasil prediksi *string* karakter per

frame dan *majority vote* pada citra pelat yang berhasil dideteksi. Metode ini dilakukan karena setiap kandidat karakter yang didapatkan tidak memiliki *ground truth*. Hasil segmentasi karakter dinilai benar 100% apabila *bounding box* hasil lokalisasi mencakup area karakter. Perhitungan performa pengenalan karakter menggunakan *majority vote* dilakukan dengan mengambil *string* karakter yang paling banyak dihasilkan dari prediksi. Hasil uji coba pada tiga waktu pengambilan yang berbeda (siang, sore, dan malam) secara berturut-turut dapat dilihat pada Tabel 5.6, Tabel 5.7, dan Tabel 5.8, sedangkan hasil uji coba waktu tiap proses dapat dilihat pada Tabel 5.9, Tabel 5.10, dan Tabel 5.11.

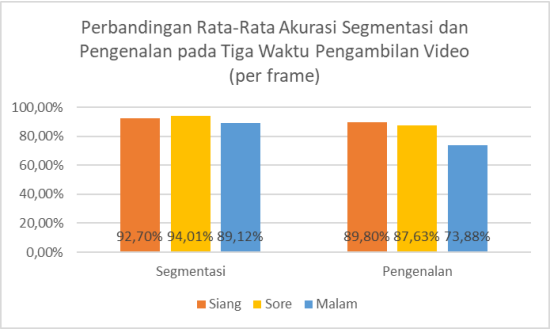
Pada siang hari didapatkan rata-rata akurasi segmentasi karakter per *frame* terbaik sebesar 99,50% pada video 2 dan rata-rata akurasi pengenalan karakter per *frame* terbaik sebesar 99,29% pada video 6. Sedangkan rata-rata akurasi pengenalan karakter hasil *majority vote* terbaik sebesar 100% didapatkan dari video 2, 3, 6, 7, 8, dan 9. Pada sore hari didapatkan rata-rata akurasi segmentasi karakter per *frame* terbaik sebesar 100% pada video 19 dan 20 serta rata-rata akurasi pengenalan karakter per *frame* terbaik sebesar 97,14% pada video 19. Sedangkan rata-rata akurasi pengenalan karakter hasil *majority vote* terbaik sebesar 100% didapatkan dari video 12, 13, 14, 18, dan 19. Terakhir, pada malam hari didapatkan rata-rata akurasi segmentasi karakter per *frame* terbaik sebesar 98,57% pada video 23 dan rata-rata akurasi pengenalan karakter per *frame* terbaik sebesar 92,74% pada video 30. Sedangkan rata-rata akurasi pengenalan karakter hasil *majority vote* terbaik sebesar 100% didapatkan dari video 24, 26, 29, dan 30.

Dari ketiga waktu pengambilan video didapatkan rata-rata akurasi segmentasi karakter per *frame* sebesar 91,94%, rata-rata akurasi pengenalan karakter per *frame* sebesar 83,77%, dan rata-rata pengenalan karakter hasil *majority vote* sebesar 88,92%. Sedangkan rata-rata waktu yang dibutuhkan untuk deteksi area pelat, segmentasi, dan pengenalan karakter pada satu pelat dalam satu *frame* masing-masing sebesar 29ms, 8ms dan 2,45ms. Rata-rata total waktu proses untuk tiap *frame* pada data video adalah

39,44ms. Grafik perbandingan keseluruhan rata-rata akurasi segmentasi karakter dan pengenalan karakter per *frame* dan *majority vote* pada tiga waktu pengambilan masing-masing dapat dilihat pada Gambar 5.4 dan Gambar 5.5.

Tabel 5.6 Perbandingan rata-rata akurasi segmentasi dan pengenalan karakter pada data video (waktu pengambilan siang)

Video	**Rata-rata Akurasi (%)**		
	Segmentasi Karakter per *frame*	**Pengenalan Karakter per *frame***	**Pengenalan Karakter *majority vote***
1	95,56	88,75	87,50
2	**99,50**	91,67	**100,00**
3	78,75	85,74	**100,00**
4	88,26	79,17	85,71
5	94,64	82,74	85,71
6	94,38	**99,29**	**100,00**
7	83,57	88,31	**100,00**
8	98,75	98,57	**100,00**
9	95,00	98,75	**100,00**
10	98,57	85,00	85,71
Rata-rata	92,70	89,80	94,46



Gambar 5.4 Grafik perbandingan rata-rata akurasi segmentasi dan pengenalan per *frame* pada data video

Tabel 5.7 Perbandingan rata-rata akurasi segmentasi dan pengenalan karakter pada data video (waktu pengambilan sore)

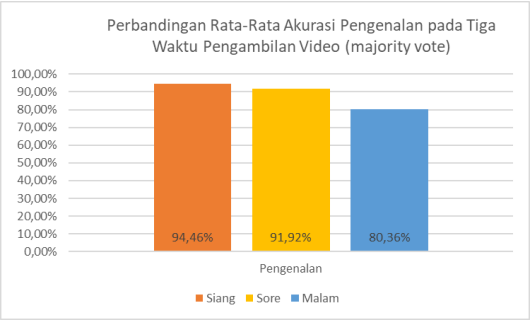
Video	**Rata-rata Akurasi (%)**		
	Segmentasi Karakter per *frame*	**Pengenalan Karakter per *frame***	**Pengenalan Karakter *majority vote***
11	99,50	91,67	88,89
12	84,63	90,19	**100,00**
13	94,03	94,88	**100,00**
14	84,29	96,31	**100,00**
15	98,13	88,29	85,71
16	99,29	75,36	85,71
17	95,76	79,11	87,50
18	82,59	96,90	**100,00**
19	**100,00**	**97,14**	**100,00**
20	**100,00**	66,43	71,43
Rata-rata	94,01	87,63	91,92

Tabel 5.8 Perbandingan rata-rata akurasi segmentasi dan pengenalan karakter pada data video (waktu pengambilan malam)

Video	**Rata-rata Akurasi (%)**		
	Segmentasi Karakter per *frame*	**Pengenalan Karakter per *frame***	**Pengenalan Karakter *majority vote***
21	77,50	59,46	57,14
22	95,17	80,80	87,50
23	**98,57**	82,00	85,71
24	92,39	89,11	**100,00**
25	97,08	82,74	85,71
26	78,89	89,65	**100,00**
27	93,13	79,91	87,50
28	64,15	0,00	0,00
29	96,88	82,38	**100,00**
30	97,50	**92,74**	**100,00**
Rata-rata	89,12	73,88	80,36

Tabel 5.9 Rata-rata waktu deteksi pelat, segmentasi, dan pengenalan karakter per *frame* pada data video (waktu pengambilan siang)

Video	Rata-rata Waktu Per *Frame* (ms)			
	Deteksi Pelat	**Segmentasi Karakter**	**Pengenalan Karakter**	**Total Waktu**
1	28,25	6,80	2,45	37,50
2	28,25	7,85	2,90	39,00
3	28,25	4,60	2,40	35,25
4	28,20	6,60	2,60	37,40
5	28,15	4,05	2,30	34,50
6	30,75	6,15	2,60	39,50
7	29,90	6,55	2,20	38,65
8	28,20	6,60	2,45	37,25
9	30,40	5,50	2,40	38,30
10	29,90	8,35	2,35	40,60
Rata-rata	29,02	6,30	2,47	37,80



Gambar 5.5 Grafik perbandingan rata-rata akurasi pengenalan *majority vote* pada data video

Tabel 5.10 Rata-rata waktu deteksi pelat, segmentasi, dan pengenalan karakter per *frame* pada data video (waktu pengambilan sore)

Video	Rata-rata Waktu Per *Frame* (ms)			
	Deteksi Pelat	**Segmentasi Karakter**	**Pengenalan Karakter**	**Total Waktu**
11	27,95	8,30	2,35	38,60
12	30,67	7,00	2,40	40,07
13	28,05	7,45	2,45	37,95
14	29,80	8,75	2,00	40,55
15	28,15	11,25	2,25	41,65
16	27,95	8,65	2,50	39,10
17	28,00	6,45	2,35	36,80
18	27,80	5,95	2,70	36,45
19	27,75	11,15	2,45	41,35
20	28,25	7,35	2,50	38,10
Rata-rata	28,44	8,23	2,40	39,06

Tabel 5.11 Rata-rata waktu deteksi pelat, segmentasi, dan pengenalan karakter per *frame* pada data video (waktu pengambilan malam)

Video	Rata-rata Waktu Per *Frame* (ms)			
	Deteksi Pelat	**Segmentasi Karakter**	**Pengenalan Karakter**	**Total Waktu**
21	29,90	8,95	2,15	41,00
22	30,20	9,45	3,25	42,90
23	29,75	10,05	2,55	42,35
24	30,28	13,60	2,40	46,28
25	29,60	9,65	2,35	41,60
26	22,40	7,40	2,05	31,85
27	29,95	11,80	2,40	44,15
28	32,20	4,33	2,62	39,15
29	30,50	10,80	2,45	43,75
30	30,50	8,56	2,56	41,63
Rata-rata	29,53	9,46	2,48	41,47

5.4.4 Uji Coba pada Data Gambar

Uji coba dilakukan untuk mengukur performa sistem yang telah dibuat pada data gambar yang berasal dari salah satu *frame* dari tiap data video yang dipilih pada saat pengendara sepeda motor berada dalam posisi berhenti. Evaluasi performa deteksi pelat, segmentasi, dan pengenalan karakter dilakukan dengan membandingkan secara visual hasil prediksi karakter pada citra pelat. Metode ini dilakukan karena setiap kandidat karakter pada pelat yang didapatkan tidak memiliki *ground truth*. Hasil lokalisasi area pelat dinilai benar 100% apabila *bounding box* hasil lokalisasi mencakup seluruh karakter pada area pelat, sedangkan hasil segmentasi karakter dinilai benar 100% apabila *bounding box* hasil lokalisasi mencakup area karakter. Hasil uji coba rata-rata akurasi pada tiga waktu pengambilan yang berbeda (siang, sore, dan malam) secara berturut-turut dapat dilihat pada Tabel 5.12, Tabel 5.13, dan Tabel 5.14, sedangkan hasil uji coba waktu tiap proses dapat dilihat pada Tabel 5.15, Tabel 5.16, dan Tabel 5.17.

Didapatkan rata-rata akurasi deteksi pelat sebesar 100% pada ketiga waktu pengambilan, rata-rata akurasi segmentasi karakter terbaik sebesar 98,75% saat pengambilan sore hari, dan rata-rata akurasi pengenalan karakter terbaik sebesar 91,55% saat pengambilan siang hari. Grafik perbandingan keseluruhan rata-rata akurasi deteksi pelat, segmentasi karakter, dan pengenalan karakter pada tiga waktu pengambilan dapat dilihat pada Gambar 5.6. Dari ketiga waktu pengambilan gambar didapatkan rata-rata akurasi deteksi pelat sebesar 100% dengan rata-rata waktu proses 18,57ms, rata-rata akurasi segmentasi karakter sebesar 97,38% dengan waktu proses 8,13ms, dan rata-rata akurasi pengenalan karakter sebesar 86,77% dengan waktu proses 2,47ms. Rata-rata total waktu proses adalah 29,17ms.

Tabel 5.12 Perbandingan akurasi deteksi pelat, segmentasi, dan pengenalan karakter pada data gambar (waktu pengambilan siang)

Gambar	Akurasi (%)		
	Deteksi Pelat	Segmentasi Karakter	Pengenalan Karakter
1	100	**100,00**	87,50
2	100	**100,00**	**100,00**
3	100	**100,00**	87,50
4	100	62,50	83,33
5	100	**100,00**	85,71
6	100	**100,00**	**100,00**
7	100	**100,00**	**100,00**
8	100	**100,00**	**100,00**
9	100	**100,00**	**100,00**
10	100	**100,00**	71,43
Rata-rata	100	96,25	91,55

Tabel 5.13 Perbandingan akurasi deteksi pelat, segmentasi, dan pengenalan karakter pada data gambar (waktu pengambilan sore)

Gambar	Akurasi (%)		
	Deteksi Pelat	Segmentasi Karakter	Pengenalan Karakter
11	100	**100,00**	**100,00**
12	100	87,50	87,50
13	100	**100,00**	**100,00**
14	100	**100,00**	**100,00**
15	100	**100,00**	**100,00**
16	100	**100,00**	71,43
17	100	**100,00**	75,00
18	100	**100,00**	**100,00**
19	100	**100,00**	**100,00**
20	100	**100,00**	71,43
Rata-rata	100	98,75	90,54

Tabel 5.14 Perbandingan akurasi deteksi pelat, segmentasi, dan pengenalan karakter pada data gambar (waktu pengambilan malam)

Gambar	Akurasi (%)		
	Deteksi Pelat	Segmentasi Karakter	Pengenalan Karakter
21	100	85,71	50,00
22	100	**100,00**	87,50
23	100	**100,00**	85,71
24	100	**100,00**	**100,00**
25	100	**100,00**	85,71
26	100	**100,00**	**100,00**
27	100	**100,00**	87,50
28	100	85,71	0,00
29	100	**100,00**	**100,00**
30	100	**100,00**	85,71
Rata-rata	100	97,14	78,21

Tabel 5.15 Waktu deteksi pelat, segmentasi, dan pengenalan karakter pada data gambar (waktu pengambilan siang)

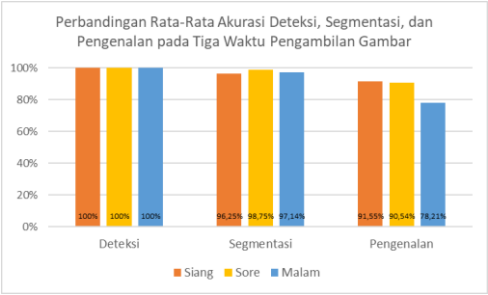
Gambar	Waktu Deteksi Pelat (ms)	Waktu Segmentasi Karakter (ms)	Waktu Pengenalan Karakter (ms)	Total Waktu (ms)
1	21,00	7,00	2,01	30,00
2	20,00	7,00	3,00	30,00
3	17,00	5,00	3,00	25,00
4	20,00	6,00	3,00	29,00
5	20,00	4,00	3,00	27,00
6	21,00	6,00	3,00	30,00
7	21,00	9,00	2,00	32,00
8	2,00	7,00	2,00	11,00
9	20,00	6,00	3,00	29,00
10	20,00	7,00	3,00	30,00
Rata-rata	18,20	6,40	2,70	27,30

Tabel 5.16 Waktu deteksi pelat, segmentasi, dan pengenalan karakter pada data gambar (waktu pengambilan sore)

Gambar	Waktu Deteksi Pelat (ms)	Waktu Segmentasi Karakter (ms)	Waktu Pengenalan Karakter (ms)	Total Waktu (ms)
11	19,00	9,00	2,00	30,00
12	20,00	6,00	2,00	28,00
13	18,00	7,00	2,00	27,00
14	19,00	8,00	3,00	30,00
15	18,00	12,00	2,00	32,00
16	20,00	9,00	3,00	32,00
17	18,00	6,00	2,00	26,00
18	20,00	6,00	2,00	28,00
19	19,00	10,00	3,00	32,00
20	20,00	7,00	2,00	29,00
Rata-rata	19,10	8,00	2,30	29,40

Tabel 5.17 Waktu deteksi pelat, segmentasi, dan pengenalan karakter pada data gambar (waktu pengambilan malam)

Gambar	Waktu Deteksi Pelat (ms)	Waktu Segmentasi Karakter (ms)	Waktu Pengenalan Karakter (ms)	Total Waktu (ms)
21	18,00	9,00	2,00	29,00
22	20,00	9,00	2,00	31,00
23	18,00	10,00	2,00	30,00
24	20,00	15,00	3,00	38,00
25	17,01	8,99	3,00	29,00
26	18,00	11,00	3,00	32,00
27	20,00	11,00	3,00	34,00
28	18,00	6,00	2,00	26,00
29	18,00	11,00	2,00	31,00
30	17,00	9,00	2,00	28,00
Rata-rata	18,40	10,00	2,40	30,80



Gambar 5.6 Grafik perbandingan rata-rata akurasi pada uji coba pada data gambar

5.5 Hasil dan Evaluasi

Pada uji coba kondisi pada citra pelat sintetis, diperoleh hasil rata-rata akurasi terbaik sebesar 97,06% pada model deteksi pelat menggunakan SSD dimana kondisi oklusi dinonaktifkan dengan waktu proses sebesar 18,57ms pada data gambar dan 29ms pada data video untuk tiap *frame*. Dari kondisi oklusi dinonaktifkan didapatkan hasil rata-rata akurasi deteksi terbaik sebesar 99% saat pengambilan video pada sore hari dan rata-rata akurasi deteksi terendah sebesar 94,19% saat malam hari. Pada siang hari, citra pelat nampak jelas walaupun beberapa gagal dideteksi karena terhalang sinar matahari seperti yang dapat dilihat pada Gambar 5.7. Sedangkan pada malam hari, citra pelat pada awal video tidak tampak jelas karena terhalang sinar lampu depan sepeda motor seperti yang dapat dilihat pada Gambar 5.8, pelat baru dapat dideteksi pada saat lampu depan sepeda motor dimatikan.

Selain kondisi pencahayaan, kualitas video juga memengaruhi hasil deteksi pelat, Gambar 5.9 merupakan salah satu contoh rusaknya *frame* video yang mengakibatkan pelat tidak dapat dideteksi. Selain keadaan dimana pelat tidak dideteksi, pada video 6, 9, dan 24 terdapat *false positive* dimana objek bukan pelat dideteksi sebagai pelat seperti yang dapat dilihat pada Gambar 5.10. Objek-objek yang terkategorikan dalam *false positive* tersebut memiliki *confidence score* antara 0,55 hingga 0,88

sedangkan objek yang terkategorikan dalam *true positive* rata-rata memiliki *confidence score* diatas 0,98 sehingga keadaan ini dapat diatasi dengan menaikkan nilai ambang batas *confidence score* suatu objek dianggap sebagai pelat.

Gambar 5.7 Citra pelat yang terkena sinar matahari

Gambar 5.8 Citra pelat yang terkena sinar lampu depan sepeda motor

Gambar 5.9 *Frame* video yang rusak

Gambar 5.10 *False positive* pada pendeteksian pelat nomor. (a) Video 6; (b) Video 9; (c) Video 24.

Pada uji coba penggantian parameter *optimizer* pada arsitektur CNN, diperoleh hasil akurasi terbaik sebesar 99,88% menggunakan *optimizer* Adam. *Optimizer* Adam tepat digunakan untuk arsitektur CNN karena menggunakan *adaptive learning rate* yang cenderung lebih cepat konvergen serta menghasilkan performa yang lebih baik dibandingkan dengan *optimizer* yang menggunakan *learning rate* statis.

Pada uji coba menggunakan data gambar pada ketiga waktu pengambilan diperoleh hasil rata-rata akurasi segmentasi karakter sebesar 97,38% dengan rata-rata waktu proses 8,13ms dan rata-rata akurasi pengenalan karakter sebesar 86,77% dengan waktu proses 2,47ms. Sedangkan pada uji coba menggunakan data video pada ketiga waktu pengambilan diperoleh hasil rata-rata akurasi segmentasi sebesar 91,94% dengan rata-rata waktu proses 8ms dan hasil pengenalan karakter terbaik menggunakan *majority vote* dibandingkan hasil rata-rata per *frame* yaitu sebesar 88,92% dengan rata-rata waktu proses 2,45ms.

Terjadi beberapa misklasifikasi dalam proses pengenalan karakter. Misklasifikasi karakter disebabkan oleh beberapa hal seperti area segmentasi yang lebih besar dari area karakter yang sebenarnya seperti yang dapat dilihat pada Gambar 5.12. Hal ini disebabkan oleh eliminasi *bounding box* menggunakan NMS yang kurang tepat seperti yang dapat dilihat pada Gambar 5.11 karena *bounding box* diprioritaskan berdasar urutan koordinat y_2 (koordinat y titik bawah kanan *bounding box*) karena output dari metode MSER tidak memiliki *confidence score* yang seharusnya digunakan dalam proses eliminasi *bounding box*.

Gambar 5.11 Contoh hasil segmentasi menggunakan MSER

Gambar 5.12 Contoh hasil segmentasi (setelah eliminasi menggunakan NMS) yang menyebabkan misklasifikasi karakter

Terdapat juga keadaan dimana karakter tidak dapat disegmentasi karena warna tulisan dengan warna dasar pelat yang kurang kontras yang terjadi saat pelat terkena sinar matahari atau lampu depan motor maupun citra pelat yang terlalu gelap seperti yang dapat dilihat pada Gambar 5.13. Selain menyebabkan gagalnya segmentasi, karakter dengan kontras rendah memiliki hasil pengenalan yang tidak tepat. Selain itu juga terjadi *false positive* dimana objek bukan karakter (seperti garis pinggiran pelat dan objek diluar pelat dikarenakan posisi pelat dalam posisi tidak lurus) tersegmentasi seperti yang dapat dilihat pada Gambar 5.14.

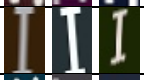
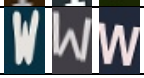
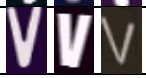
Misklasifikasi juga disebabkan kondisi tulisan pada pelat yang kurang bagus (pudar, tebal tipis tulisan tidak rata, atau tulisan terputus) dan jenis *font* pada pelat dan data latih yang berbeda seperti yang dapat dilihat pada Tabel 5.18. Hal yang dapat dilakukan untuk mengatasi permasalahan ini adalah melakukan komposisi citra sintetis dan citra asli.

Dari uji coba pada data gambar dan data video, didapatkan bahwa waktu pengambilan sore hari memperoleh rata-rata akurasi deteksi area pelat, segmentasi, dan pengenalan karakter tertinggi dibandingkan waktu pengambilan siang dan malam. Hal ini karena pada sore hari,sinar matahari tidak terlalu terang dan lampu depan sepeda motor tidak menghalangi sehingga baik citra pelat maupun karakter di dalamnya terlihat jelas.

Gambar 5.13 Contoh karakter dengan kontras rendah

Gambar 5.14 *False positive* pada segmentasi karakter

Tabel 5.18 Kondisi tulisan yang menyebabkan misklasifikasi

Karakter pada Pelat	**Kelas Aktual**	**Kelas Prediksi**	**Karakter Aktual pada Dataset**	**Karakter Prediksi pada Dataset**
	4	Z		
	H	N		
	I	1		
	W	V		

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa yang akan datang.

6.1 Kesimpulan

Dalam pengerjaan Tugas Akhir ini setelah melalui tahap perancangan aplikasi, implementasi metode, serta uji coba, diperoleh kesimpulan sebagai berikut:

1. Berdasarkan uji coba kondisi pada citra pelat sintetis, model deteksi area pelat menggunakan SSD menghasilkan rata-rata akurasi terbaik pada data video sebesar 97,06% dengan rata-rata waktu proses per *frame* 29ms sedangkan pada data gambar sebesar 100% dengan rata-rata waktu proses 18,57ms didapatkan pada saat kondisi oklusi pada dataset pelat sintetis dinonaktifkan.
2. Berdasarkan uji coba parameter *optimizer* pada arsitektur CNN untuk model pengenalan karakter didapatkan akurasi terbaik sebesar 99,88% menggunakan Adam *optimizer*.
3. Metode *Maximally Stable Extremal Regions* untuk segmentasi karakter pada pelat nomor berhasil dilakukan dengan akurasi rata-rata sebesar 91,94% pada data video dengan rata-rata waktu proses per *frame* 8ms dan sebesar 97,38% pada data gambar dengan rata-rata waktu proses 8,13ms.
4. Sistem pengenalan pelat nomor kendaraan dengan data latih berupa citra sintetis telah berhasil diimplementasikan menggunakan CNN dengan rata-rata akurasi pengujian sebesar 83,77% pada data video per

frame, 88,92% pada data video hasil *majority vote* dengan rata-rata waktu proses per *frame* 2,45ms dan 86,77% pada data gambar dengan rata-rata waktu proses 2,47ms. Rata-rata total waktu proses tiap *frame* pada data video adalah 39,44ms dan 29,17ms pada data gambar.

5. Kondisi pencahayaan pada tiga waktu pengambilan data uji memengaruhi hasil uji coba dimana didapatkan rata-rata akurasi deteksi area pelat, segmentasi, dan pengenalan karakter terbaik pada sore hari karena intensitas cahaya yang bagus dibandingkan waktu pengambilan siang dan malam hari.

6.2 Saran

Saran yang diberikan untuk pengembangan sistem pengenalan pelat nomor kendaraan menggunakan *Convolutional Neural Network* pada data video, yaitu:

1. Menambah kondisi pada pembuatan citra sintetis.
2. Pengembangan sistem yang dapat melakukan segmentasi dan pengenalan karakter pada kondisi kontras rendah.
3. Melakukan uji coba komposisi dataset citra asli dan sintetis pada proses pelatihan.
4. Melakukan eksplorasi parameter selain *optimizer* yang dapat menambah performa arsitektur CNN seperti *learning rate*, *activation function*, jumlah dan *stride filter* konvolusi, atau ukuran *max pooling*.

DAFTAR PUSTAKA

[1] D. Menotti, G. Chiachia, A. X. Falcão dan V. Oliveira Neto, "Vehicle License Plate Recognition With Random Convolutional Networks," *Brazilian Symposium of Computer Graphic and Image Processing*, pp. 298-303, 2014.

[2] M. M. S. Rahman, M. S. Nasrin, M. Mostakim dan M. Z. Alom, "Bangla License Plate Recognition Using Convolutional Neural Networks (CNN)," 2018.

[3] T. Björklund, A. Fiandrotti, M. Annarumma, G. Francini dan E. Magli, "Robust License Plate Recognition using Neural Networks Trained on Synthetic Images," *Pattern Recognition*, vol. 93, pp. 134-146, 2019.

[4] T. Sutoyo dan E. Mulyanto, Teori Pengolahan Citra Digital, Yogyakarta: Andi, 2009.

[5] C. S. Hsu dan S. F. Tu, "An Imperceptible Watermarking Scheme Using Variation and Modular Operations," *International Journal of Hybrid Information Technology*, vol. 1, pp. 9-16, 2008.

[6] R. Munir, Pengolahan Citra Digital dengan Pendekatan Algoritmik, Bandung: Informatika, 2004.

[7] R. Sigit, A. Basuki, N. Ramadijanti dan D. Pramadihanto, Step by step Pengolahan Citra Digital, Jogjakarta: Andi, 2006.

[8] Y. A. Hambali, "Aplikasi Area Process Berbasis C#," 2011. [Online]. Available: https://ilmukomputer.org/wp-content/uploads/2011/03/AreaProcess_YudiAhmadH.pdf. [Diakses 4 Desember 2019].

[9] B. Yuwono, "Image Smoothing Menggunakan MidPoint Filtering, Median Filtering, Modus Filtering, dan

Gaussian Filtering," *Telematika,* vol. 7, pp. 65-75, 2010.

[10] A. Balza dan F. Kartika, Teknik Pengolahan Citra. Digital Menggunakan Delphi, Yogyakarta: Ardi Publishing, 2005.

[11] S. Hadi, "Pengolahan Citra Dijital - Situs Setiawan Hadi," [Online]. Available: http://setiawanhadi.unpad.ac.id/Welcome%20to%20Setiawan%20Hadi%20homepage_files/pendidikan/Ganjil1516/CG/GK05_Transformasi%203D.pptx. [Diakses 23 Desember 2019].

[12] C. Patel, D. Shah dan A. Patel, "Automatic Number Plate Recognition System (ANPR): A Survey," *International Journal of Computer Applications,* vol. 69, pp. 21-23, 2013.

[13] S. Fadillah, "Penerapan Pengolahan Citra menggunakan Metode Deep Learning untuk Mendeteksi Kecacatan Permukaan Buah Manggis," Yogyakarta, 2017.

[14] D. Hubel dan T. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *Journal of Physiology,* vol. 195, p. 215–243, 1968.

[15] M. Zufar dan B. Setiyono, "Convolutional Neural Networks untuk Pengenalan Wajah Secara Real-Time," *Jurnal Sains dan Seni ITS,* vol. 5, pp. 2337-3520, 2016.

[16] "Convolutional Neural Network," MathWorks, [Online]. Available: https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html. [Diakses 28 November 2019].

[17] "Deep learning for complete beginners: convolutional neural networks with keras," Cambridgespark, 20 March 2017. [Online]. Available: https://cambridgespark.com/content/tutorials/convolut

ional-neural-networks-with-keras/index.html. [Diakses 28 November 2019].

[18] "An Intuitive Explanation of Convolutional Neural Networks," Ujjwalkarn, 11 Agustus 2016. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets. [Diakses 29 November 2019].

[19] I. W. Suartika, A. Y. Wijaya dan R. Soelaiman, "Klasifikasi Citra Menggunakan Convolutional pada Caltech 101," *JURNAL TEKNIK ITS*, vol. 5, 2016.

[20] S. Sena, "Pengenalan Deep Learning Neural Network," 28 October 207. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac. [Diakses 30 November 2019].

[21] G. Hinton, *Neural Networks for Machine Learning.*

[22] S. Ruder, "Ruder.io," 19 Januari 2016. [Online]. Available: http://ruder.io/optimizing-gradient-descent/index.html#rmsprop. [Diakses 2 Desember 2019].

[23] A. Budhiraja, "Dropout in (Deep) Machine Learning," [Online]. Available: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5. [Diakses 28 November 2019].

[24] A. Karpathy, "Convolutional Neural Networks for Visual Recognition," Stanford University, [Online]. Available: http://cs231n.github.io/. [Diakses 30 November 2019].

[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu dan A. C. Berg, "SSD: Single Shot MultiBox Detector," 2016.

[26] K. Simonyan dan A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014.

[27] A. Tabassum dan S. A. Dhondse, "Text Detection Using MSER and Stroke Width Transform," *2015 Fifth International Conference on Communication Systems and Network Technologies,* pp. 568-571, 2015.

[28] M. Donoser dan H. Bischof, "Efficient Maximally Stable Extremal Region (MSER) Tracking," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06),* pp. 553-560, 2006.

[29] J. Matas, O. Chum, M. Urban dan T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing,* vol. 22, no. 10, pp. 761-767, 2004.

[30] P. F. Felzenszwalb, R. B. Girshick, D. McAllester dan D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," *IEEE transactions on pattern analysis and machine intelligence,* pp. 1627-1645, 2010.

[31] A. Rosebrock, "(Faster) Non-Maximum Suppression in Python," 16 Februari 2015. [Online]. Available: https://www.pyimagesearch.com/2015/02/16/faster-non-maximum-suppression-python/. [Diakses 3 Desember 2019].

[32] N. R. Radliya, "Pertemuan 3 - Materi [dm - 2016]," 3 April 2017. [Online]. Available: https://repository.unikom.ac.id/49055/1/Pertemuan%203%20-%20Materi%20%5BDM%20-%202016%5D.pdf. [Diakses 9 Juni 2019].

[33] M. Sokolova dan G. Lapalme, "A systematic analysis of performance measures for classification tasks," vol. 45, no. 4, pp. 427-437, 2009.

[34] "About Python," Python, [Online]. Available: https://www.python.org/about/. [Diakses 28 November 2019].

[35] "Keras: The Python Deep Learning library," Keras, [Online]. Available: https://keras.io/. [Diakses 27 November 2019].

[36] "TensorFlow," TensorFlow, [Online]. Available: https://www.tensorflow.org/. [Diakses 27 November 2019].

[37] "OpenCV," [Online]. Available: https://opencv.org/. [Diakses 27 November 2019].

[38] "Python Imaging Library (PIL)," PythonWare, [Online]. Available: http://www.pythonware.com/products/pil/. [Diakses 24 November 2019].

[39] "NumPy," NumPy, [Online]. Available: http://www.numpy.org/. [Diakses 27 November 2019].

[40] "Scikit-learn," Scikit-learn, [Online]. Available: http://scikit-learn.org/stable/index.html. [Diakses 27 November 2019].

[41] "Matplotlib," Matplotlib, [Online]. Available: https://matplotlib.org/index.html. [Diakses 27 November 2019].

[42] J. Krol, B. J. D'mello dan M. Satheesh, Web Development with MongoDB and NodeJS - Second Edition, Birmingham: Packt Publishing, 2015.

[43] "Express - Node.js web application framework," 2017. [Online]. Available: https://expressjs.com/. [Diakses 8 Desember 2019].

[44] Y. Zhangling dan D. Mao, "A Real-Time Group Communication Architecture Based on WebSocket," *International Journal of Computer and Communication Engineering,* vol. 1, 2012.

[45] "Introduction to Redis," [Online]. Available: https://redis.io/topics/introduction. [Diakses 7 Desember 2019].

[46] J. Nelson, “Focused Topics in Redis Day 2 - #5,” 2016. [Online]. Available: http://intro2libsys.com/focused-redis-topics/day-two/pubsub. [Diakses 7 Desember 2019].

[47] B. Buku Pintar Framework Yii, Yogyakarta: Mediakom, 2013.

[48] M. J. Huiskes dan M. S. Lew, “The MIR Flickr Retrieval Evaluation,” dalam *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, ACM, 2008.

[49] “The MIRFLICKR Retrieval Evaluation,” [Online]. Available: https://press.liacs.nl/mirflickr/#sec_publications. [Diakses 27 November 2019].

[50] “Tanda nomor kendaraan bermotor,” 24 November 2019. [Online]. Available: https://id.wikipedia.org/wiki/Tanda_nomor_kendaraan_bermotor. [Diakses 27 November 2019].

[51] B. Ramadan, “Indonesia License Plate font,” 22 April 2013. [Online]. Available: https://www.dafont.com/indonesia-license-plate.font. [Diakses 2019 Juni 30].

[52] T. Björklund, “License Plate Recognition using Convolutional Neural Networks Trained on Synthetic Images,” 2018.

LAMPIRAN

L.1 Hasil Uji Coba Semua Kondisi

Siang	Akurasi	Sore	Akurasi	Malam	Akurasi
Video 1	100,00%	Video 11	100,00%	Video 21	90,00%
Video 2	95,00%	Video 12	100,00%	Video 22	85,00%
Video 3	100,00%	Video 13	80,00%	Video 23	100,00%
Video 4	95,00%	Video 14	90,00%	Video 24	0,00%
Video 5	100,00%	Video 15	100,00%	Video 25	95,00%
Video 6	95,00%	Video 16	95,00%	Video 26	70,00%
Video 7	70,00%	Video 17	85,00%	Video 27	100,00%
Video 8	100,00%	Video 18	95,00%	Video 28	69,23%
Video 9	95,00%	Video 19	100,00%	Video 29	100,00%
Video 10	100,00%	Video 20	100,00%	Video 30	95,00%
Rata-rata	95,00%	Rata-rata	94,50%	Rata-rata	80,42%

L.2 Hasil Uji Coba Tanpa Oklusi

Siang	Akurasi	Sore	Akurasi	Malam	Akurasi
Video 1	100,00%	Video 11	100,00%	Video 21	90,00%
Video 2	100,00%	Video 12	100,00%	Video 22	100,00%
Video 3	100,00%	Video 13	100,00%	Video 23	100,00%
Video 4	100,00%	Video 14	90,00%	Video 24	95,00%
Video 5	100,00%	Video 15	100,00%	Video 25	100,00%
Video 6	95,00%	Video 16	100,00%	Video 26	80,00%
Video 7	95,00%	Video 17	100,00%	Video 27	100,00%
Video 8	100,00%	Video 18	100,00%	Video 28	76,92%
Video 9	90,00%	Video 19	100,00%	Video 29	100,00%
Video 10	100,00%	Video 20	100,00%	Video 30	100,00%
Rata-rata	98,00%	Rata-rata	99,00%	Rata-rata	94,19%

L.3 Hasil Uji Coba Tanpa Bayangan

Siang	Akurasi	Sore	Akurasi	Malam	Akurasi
Video 1	100,00%	Video 11	0,00%	Video 21	90,00%
Video 2	25,00%	Video 12	60,00%	Video 22	75,00%

Siang	Akurasi	Sore	Akurasi	Malam	Akurasi
Video 3	70,00%	Video 13	35,00%	Video 23	100,00%
Video 4	85,00%	Video 14	90,00%	Video 24	0,00%
Video 5	100,00%	Video 15	100,00%	Video 25	30,00%
Video 6	95,00%	Video 16	100,00%	Video 26	40,00%
Video 7	85,00%	Video 17	60,00%	Video 27	100,00%
Video 8	100,00%	Video 18	90,00%	Video 28	53,85%
Video 9	95,00%	Video 19	100,00%	Video 29	95,00%
Video 10	95,00%	Video 20	100,00%	Video 30	100,00%
Rata-rata	85,00%	Rata-rata	73,50%	Rata-rata	68,38%

L.4 Hasil Uji Coba Tanpa Transparansi

Siang	Akurasi	Sore	Akurasi	Malam	Akurasi
Video 1	100,00%	Video 11	55,00%	Video 21	90,00%
Video 2	95,00%	Video 12	100,00%	Video 22	90,00%
Video 3	100,00%	Video 13	100,00%	Video 23	100,00%
Video 4	100,00%	Video 14	90,00%	Video 24	75,00%
Video 5	100,00%	Video 15	100,00%	Video 25	75,00%
Video 6	95,00%	Video 16	100,00%	Video 26	65,00%
Video 7	90,00%	Video 17	100,00%	Video 27	100,00%
Video 8	100,00%	Video 18	90,00%	Video 28	53,85%
Video 9	100,00%	Video 19	100,00%	Video 29	95,00%
Video 10	95,00%	Video 20	100,00%	Video 30	100,00%
Rata-rata	97,50%	Rata-rata	93,50%	Rata-rata	84,38%

L.5 Hasil Uji Coba Tanpa Noise pada Pelat

Siang	Akurasi	Sore	Akurasi	Malam	Akurasi
Video 1	100,00%	Video 11	95,00%	Video 21	90,00%
Video 2	100,00%	Video 12	100,00%	Video 22	100,00%
Video 3	100,00%	Video 13	100,00%	Video 23	100,00%
Video 4	100,00%	Video 14	90,00%	Video 24	100,00%
Video 5	100,00%	Video 15	100,00%	Video 25	100,00%
Video 6	95,00%	Video 16	100,00%	Video 26	70,00%
Video 7	90,00%	Video 17	100,00%	Video 27	100,00%

Siang	Akurasi	Sore	Akurasi	Malam	Akurasi
Video 8	100,00%	Video 18	100,00%	Video 28	53,85%
Video 9	100,00%	Video 19	100,00%	Video 29	100,00%
Video 10	100,00%	Video 20	100,00%	Video 30	100,00%
Rata-rata	98,50%	Rata-rata	98,50%	Rata-rata	91,38%

L.6 Hasil Uji Coba Tanpa Motion Blur pada Pelat

Siang	Akurasi	Sore	Akurasi	Malam	Akurasi
Video 1	100,00%	Video 11	90,00%	Video 21	90,00%
Video 2	85,00%	Video 12	86,67%	Video 22	45,00%
Video 3	100,00%	Video 13	50,00%	Video 23	100,00%
Video 4	65,00%	Video 14	85,00%	Video 24	100,00%
Video 5	100,00%	Video 15	100,00%	Video 25	95,00%
Video 6	95,00%	Video 16	100,00%	Video 26	60,00%
Video 7	85,00%	Video 17	100,00%	Video 27	100,00%
Video 8	100,00%	Video 18	100,00%	Video 28	76,92%
Video 9	100,00%	Video 19	100,00%	Video 29	100,00%
Video 10	100,00%	Video 20	100,00%	Video 30	70,00%
Rata-rata	93,00%	Rata-rata	91,17%	Rata-rata	83,69%

L.7 Hasil Uji Coba Tanpa Noise Keseluruhan

Siang	Akurasi	Sore	Akurasi	Malam	Akurasi
Video 1	100,00%	Video 11	30,00%	Video 21	90,00%
Video 2	90,00%	Video 12	100,00%	Video 22	100,00%
Video 3	100,00%	Video 13	100,00%	Video 23	100,00%
Video 4	100,00%	Video 14	90,00%	Video 24	80,00%
Video 5	100,00%	Video 15	100,00%	Video 25	95,00%
Video 6	95,00%	Video 16	100,00%	Video 26	70,00%
Video 7	95,00%	Video 17	100,00%	Video 27	100,00%
Video 8	100,00%	Video 18	100,00%	Video 28	76,92%
Video 9	100,00%	Video 19	100,00%	Video 29	100,00%
Video 10	100,00%	Video 20	100,00%	Video 30	100,00%
Rata-rata	98,00%	Rata-rata	92,00%	Rata-rata	91,19%

L.8 Hasil Uji Coba Tanpa Motion Blur Keseluruhan

Siang	Akurasi	Sore	Akurasi	Malam	Akurasi
Video 1	100,00%	Video 11	65,00%	Video 21	90,00%
Video 2	100,00%	Video 12	93,33%	Video 22	100,00%
Video 3	100,00%	Video 13	95,00%	Video 23	100,00%
Video 4	100,00%	Video 14	90,00%	Video 24	100,00%
Video 5	100,00%	Video 15	100,00%	Video 25	95,00%
Video 6	95,00%	Video 16	100,00%	Video 26	70,00%
Video 7	90,00%	Video 17	100,00%	Video 27	100,00%
Video 8	100,00%	Video 18	100,00%	Video 28	92,31%
Video 9	90,00%	Video 19	100,00%	Video 29	100,00%
Video 10	100,00%	Video 20	100,00%	Video 30	100,00%
Rata-rata	97,50%	Rata-rata	94,33%	Rata-rata	94,73%

L.9 Hasil Uji Coba Tanpa Blur

Siang	Akurasi	Sore	Akurasi	Malam	Akurasi
Video 1	100,00%	Video 11	55,00%	Video 21	90,00%
Video 2	100,00%	Video 12	73,33%	Video 22	20,00%
Video 3	100,00%	Video 13	30,00%	Video 23	100,00%
Video 4	95,00%	Video 14	85,00%	Video 24	40,00%
Video 5	100,00%	Video 15	90,00%	Video 25	95,00%
Video 6	95,00%	Video 16	100,00%	Video 26	60,00%
Video 7	85,00%	Video 17	60,00%	Video 27	100,00%
Video 8	100,00%	Video 18	100,00%	Video 28	61,54%
Video 9	95,00%	Video 19	100,00%	Video 29	95,00%
Video 10	95,00%	Video 20	100,00%	Video 30	45,00%
Rata-rata	96,50%	Rata-rata	79,33%	Rata-rata	70,65%

L.10 Hasil Uji Coba Optimizer Adagrad

Kelas	Precision	Recall	F1-Score	Support
0	0,9979	0,9979	0,9979	478
1	1,0000	1,0000	1,0000	442
2	0,9977	1,0000	0,9989	437
3	1,0000	1,0000	1,0000	415

Kelas	Precision	Recall	F1-Score	Support
4	0,9956	1,0000	0,9978	451
5	0,9978	0,9978	0,9978	446
6	1,0000	0,9978	0,9989	452
7	0,9978	1,0000	0,9989	458
8	1,0000	1,0000	1,0000	477
9	1,0000	1,0000	1,0000	460
A	0,9977	1,0000	0,9989	441
B	1,0000	0,9979	0,9989	475
C	1,0000	0,9978	0,9989	451
D	0,9978	0,9978	0,9978	465
E	1,0000	0,9952	0,9976	417
F	0,9936	1,0000	0,9968	465
G	0,9977	0,9955	0,9966	445
H	1,0000	1,0000	1,0000	458
I	1,0000	0,9954	0,9977	435
J	0,9953	0,9977	0,9965	428
K	1,0000	0,9978	0,9989	447
L	1,0000	1,0000	1,0000	455
M	1,0000	1,0000	1,0000	445
N	0,9976	1,0000	0,9988	417
O	1,0000	1,0000	1,0000	425
P	0,9979	0,9979	0,9979	487
Q	0,9957	1,0000	0,9978	458
R	1,0000	1,0000	1,0000	457
S	0,9905	0,9976	0,9941	420
T	1,0000	1,0000	1,0000	427
U	1,0000	0,9958	0,9979	472
V	1,0000	1,0000	1,0000	461
W	1,0000	0,9956	0,9978	459
X	0,9958	1,0000	0,9979	470
Y	1,0000	0,9977	0,9989	436
Z	1,0000	0,9936	0,9968	468

Kelas	Precision	Recall	F1-Score	Support
Weighted Average	0,9985	0,9985	0,9985	16200

L.11 Hasil Uji Coba Optimizer SGD

Kelas	Precision	Recall	F1-Score	Support
0	0,0597	0,0502	0,0545	478
1	0,0000	0,0000	0,0000	442
2	0,0000	0,0000	0,0000	437
3	0,0000	0,0000	0,0000	415
4	0,0000	0,0000	0,0000	451
5	0,1659	0,3946	0,2336	446
6	0,0000	0,0000	0,0000	452
7	0,1564	0,3428	0,2148	458
8	0,0000	0,0000	0,0000	477
9	0,0000	0,0000	0,0000	460
A	0,0000	0,0000	0,0000	441
B	0,0447	0,8421	0,0849	475
C	0,0000	0,0000	0,0000	451
D	0,0887	0,2774	0,1344	465
E	0,1667	0,0024	0,0047	417
F	0,0000	0,0000	0,0000	465
G	0,0270	0,0022	0,0041	445
H	0,0000	0,0000	0,0000	458
I	0,8667	0,1195	0,2101	435
J	0,0000	0,0000	0,0000	428
K	0,0000	0,0000	0,0000	447
L	0,0000	0,0000	0,0000	455
M	0,2340	0,0247	0,0447	445
N	0,0000	0,0000	0,0000	417
O	0,3101	0,9435	0,4668	425
P	0,0000	0,0000	0,0000	487
Q	0,0000	0,0000	0,0000	458
R	0,0000	0,0000	0,0000	457

Kelas	Precision	Recall	F1-Score	Support
S	0,0000	0,0000	0,0000	420
T	0,0000	0,0000	0,0000	427
U	0,0000	0,0000	0,0000	472
V	0,0000	0,0000	0,0000	461
W	0,0000	0,0000	0,0000	459
X	0,2385	0,0553	0,0898	470
Y	0,0000	0,0000	0,0000	436
Z	0,1412	0,5150	0,2216	468
Weighted Average	0,0685	0,0999	0,0488	16200

L.12 Hasil Uji Coba Optimizer Adam

Kelas	Precision	Recall	F1-Score	Support
0	1,0000	0,9979	0,9990	478
1	1,0000	1,0000	1,0000	442
2	0,9977	1,0000	0,9989	437
3	1,0000	1,0000	1,0000	415
4	0,9978	1,0000	0,9989	451
5	1,0000	0,9955	0,9978	446
6	0,9956	0,9978	0,9967	452
7	0,9978	1,0000	0,9989	458
8	0,9979	1,0000	0,9990	477
9	0,9978	0,9978	0,9978	460
A	1,0000	1,0000	1,0000	441
B	1,0000	0,9979	0,9989	475
C	0,9978	1,0000	0,9989	451
D	0,9979	1,0000	0,9989	465
E	1,0000	0,9952	0,9976	417
F	0,9979	1,0000	0,9989	465
G	0,9978	0,9978	0,9978	445
H	1,0000	1,0000	1,0000	458
I	0,9977	0,9977	0,9977	435
J	0,9977	1,0000	0,9988	428

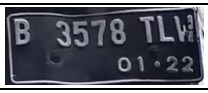
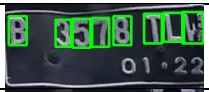
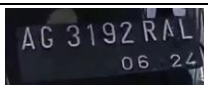
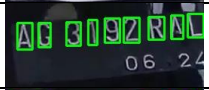
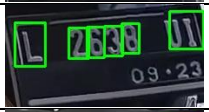
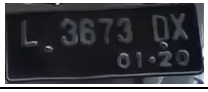
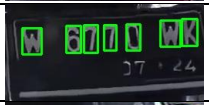
Kelas	Precision	Recall	F1-Score	Support
K	1,0000	0,9978	0,9989	447
L	1,0000	1,0000	1,0000	455
M	1,0000	1,0000	1,0000	445
N	0,9976	1,0000	0,9988	417
O	1,0000	1,0000	1,0000	425
P	0,9980	1,0000	0,9990	487
Q	1,0000	1,0000	1,0000	458
R	0,9978	1,0000	0,9989	457
S	0,9976	1,0000	0,9988	420
T	1,0000	1,0000	1,0000	427
U	1,0000	0,9958	0,9979	472
V	0,9978	1,0000	0,9989	461
W	1,0000	0,9956	0,9978	459
X	1,0000	1,0000	1,0000	470
Y	1,0000	1,0000	1,0000	436
Z	1,0000	0,9936	0,9968	468
Weighted Average	0,9989	0,9989	0,9989	16200

L.13 Hasil Uji Coba Optimizer RMSprop

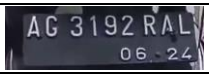
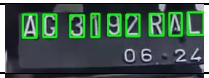
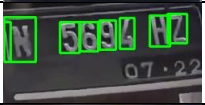
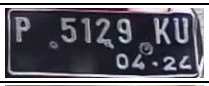
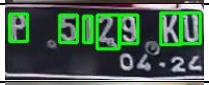
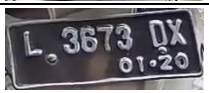
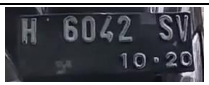
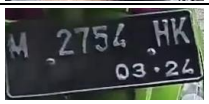
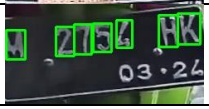
Kelas	Precision	Recall	F1-Score	Support
0	0,9979	0,9958	0,9969	478
1	1,0000	1,0000	1,0000	442
2	1,0000	1,0000	1,0000	437
3	1,0000	1,0000	1,0000	415
4	0,9978	0,9978	0,9978	451
5	1,0000	0,9955	0,9978	446
6	0,9956	0,9978	0,9967	452
7	0,9978	1,0000	0,9989	458
8	0,9979	0,9979	0,9979	477
9	0,9978	0,9978	0,9978	460
A	1,0000	1,0000	1,0000	441
B	1,0000	0,9979	0,9989	475

Kelas	Precision	Recall	F1-Score	Support
C	0,9978	0,9978	0,9978	451
D	0,9915	0,9978	0,9946	465
E	1,0000	0,9952	0,9976	417
F	0,9957	1,0000	0,9979	465
G	0,9978	0,9978	0,9978	445
H	1,0000	1,0000	1,0000	458
I	1,0000	1,0000	1,0000	435
J	0,9953	0,9953	0,9953	428
K	1,0000	0,9978	0,9989	447
L	1,0000	1,0000	1,0000	455
M	1,0000	1,0000	1,0000	445
N	0,9976	1,0000	0,9988	417
O	1,0000	1,0000	1,0000	425
P	0,9980	1,0000	0,9990	487
Q	0,9957	1,0000	0,9978	458
R	1,0000	0,9978	0,9989	457
S	0,9952	0,9976	0,9964	420
T	1,0000	1,0000	1,0000	427
U	1,0000	0,9958	0,9979	472
V	1,0000	1,0000	1,0000	461
W	1,0000	0,9978	0,9989	459
X	0,9979	1,0000	0,9989	470
Y	0,9977	1,0000	0,9989	436
Z	1,0000	0,9936	0,9968	468
Weighted Average	0,9985	0,9985	0,9985	16200

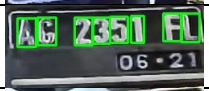
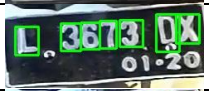
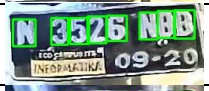
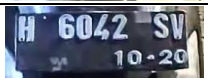
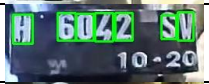
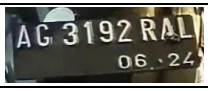
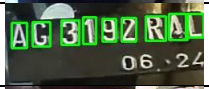
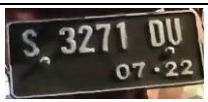
L.14 Hasil Uji Coba pada Data Gambar Waktu Pengambilan Siang

Gambar	Hasil Lokalisasi Pelat	Hasil Segmentasi Karakter	Hasil Pengenalan Karakter
1			B3578TLV
2			AG3192RAL
3			6G2351FL
4			L26380
5			L3673QX
6			L6382ML
7			W6770WK
8			L4025HL
9			AE3395WR
10			V3440QJ

L.15 Hasil Uji Coba pada Data Gambar Waktu Pengambilan Sore

Gambar	Hasil Lokalisasi Pelat	Hasil Segmentasi Karakter	Hasil Pengenalan Karakter
11			AG3192RAL
12			XN5694HZ
13			L3669TT
14			P5129KU
15			W3440WJ
16			L36T3QX
17			HG235OFL
18			H6042SV
19			P2083WN
20			H2N54HK

L.16 Hasil Uji Coba pada Data Gambar Waktu Pengambilan Malam

Gambar	Hasil Lokalisasi Pelat	Hasil Segmentasi Karakter	Hasil Pengenalan Karakter
21			WO66RB
22			A62351FL
23			L3673QX
24			N3526NBB
25			N6042SV
26			AG3192RAL
27			AA4427M6
28			N8B8N8
29			S3271DU
30			L36691T

L.17 Hasil Uji Coba Majority Vote pada Data Video

Video	Majority Vote	Video	Majority Vote
1	B3578TLV	16	L36T3DX
2	AG3192RAL	17	RG2351FL
3	2351F	18	H6042SV
4	L2638U1	19	P2083WN
5	L3673QX	20	M2N5ZHK
6	L6382ML	21	W3O66RN
7	W6770WK	22	A62351FL
8	L4025HL	23	L36730X
9	AE3395WR	24	N3526NBB
10	W3440QJ	25	N6042SV
11	AG3392RAL	26	AG3192RAL
12	N5694HZ	27	AA4427M6
13	L3669TT	28	NB8N8
14	P5129KU	29	S3271DU
15	W3440QJ	30	L3669TT

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Ghifaroza Rahmadiana, lahir di Surabaya pada tanggal 31 Juli 1998. Penulis menempuh pendidikan mulai dari TK Ar-Rahmah (2002-2004), SD Islam Maryam (2004-2010), SMP Negeri 6 Surabaya (2010-2013), SMA Negeri 5 Surabaya (2013-2016), dan saat ini sedang menjalani pendidikan S1 Teknik Informatika di Institut Teknologi Sepuluh Nopember. Penulis aktif dalam organisasi dan kepanitiaan Himpunan Mahasiswa Teknik Computer (HMTC) dan Schematics diantaranya menjadi staf Departemen Kesejahteraan Mahasiswa HMTC ITS 2017-2018, staf ahli Departemen Kesejahteraan Mahasiswa HMTC ITS 2018-2019, Badan Pengurus Harian III Biro Web dan Kesekretariatan Schematics ITS 2017 dan Badan Pengurus Harian I Biro Web dan Kesekretariatan Schematics ITS 2018. Komunikasi dengan penulis dapat melalui telepon: +6287878784431 dan *e-mail*: **ghifarozarahmadiana@gmail.com**.